Sapienza - University of Rome

Master in Engineering In Computer Science

Neural Netoworks course

Revisiting the paper "A Neural algorithm of Artistic style"

Presented by:

Giovanni Regina, Riccardo Viviano

Contents

1 Introduction

This work is a revisit of the Paper "A Neural algorithm of Artistic style" [1], made by Riccardo Viviano and Giovanni Regina. Convolutional networks have recently enjoyed a great success in large-scale image recognition, which has become possible due to the large public image repositories, such as ImageNet [5] , high-performance computing systems, such as GPUs, and the cloud computing [4]. The authors of [3] exploit the structure of the CNN, in particolar the VGG19 [4], to mix the content of an image (like a picture of a building, etc..) with the style of another image (like a portrait or whatever) obtaining a result that makes the former looks like a portrait. We have implemented the network described in the paper, with the same architecture and the same settings on the Google Colab platform, that gives the fast (cloud) computational power needed for Deep Neural Network's calculus. Then we made some adjustments, like the introduction of the Total Variation Loss, and we made different tries varying the hyperparameters used during the training of our model. The rest of the paper is organised as follows: In section 2 we explore the architecture of the VGG19 network, in order to understand why the authors have made their choices, and also to change the settings in order to achieve better results. In section 3 we introduce the mathematical formulas that the model computes during the training step. Finally in the last section are showed the results according to the different settings, with the respective analysis.

2 The VGG Network

The VGG19 is a network that empower the Deep Convolutional Neural Networks's architecture to make high accurate predictions on the object recognition task, even using small convolutional filters (3x3) [4]. The achievement is reached by building a Deep Network with 19 hidden layers (originally they were 11, then increased to achieve better performances to 13, 16 and finally 19), trained on the Imagenet dataset , but scalable to other datasets. The network architecture is composed by convolutional layers distributed in 5 levels plus 3 fully connected layers as shown in Fig. 1, in this way: -First Level: Take as input an RGB image (224x224) and apply 64 conv. filters twice, generating in this way the first 2 layers ($conv1_1, conv1_2$). Then the max pooling phase that makes the shape of the feature maps of 112x112x64. -Second Level: Apply 128 conv. filters twice, generating in this way other 2 layers ($conv2_1, conv2_2$) that, after the max pooling phase, will have both the shape of 56x56x128. -Third Level: 256 conv. filters four times (originally they were 2) to generate 4 other layers ($conv3_1, conv3_2, conv3_3, conv3_4$), followed by the max pooling phase, for a final shape of 28x28x256. -Fourth Level: 512 conv. filters four times (originally they were 2) to generate 4 other layers ($conv4_1, conv4_2, conv4_3, conv4_4$), max pooling for a shape of 14x14x512. -Fifth Layer: 512 conv. filters four times (originally they were 2) to generate 4 other layers ($conv3_1, conv3_2, conv3_3, conv3_4$), followed by the same max pooling phase of each level, for a final shape of 7x7x512. The total number of training parameters (weights and biases) are around 144 millions (without the 3 fully connected layers). The authors of [3] noted that in the VGG Network, but the same reasoning can be applied to other Convolutional Neural Networks trained on image recognition, along the processing hierarchy of the network, the input image is transformed into representations that increasingly care more about the actual content of the image rather than his definition, like the exact pixel values of the input image (what the lower layers instead do), and in the other hand, the style of an input image can be obtained by taking care of different layers' correlation (see Methods).

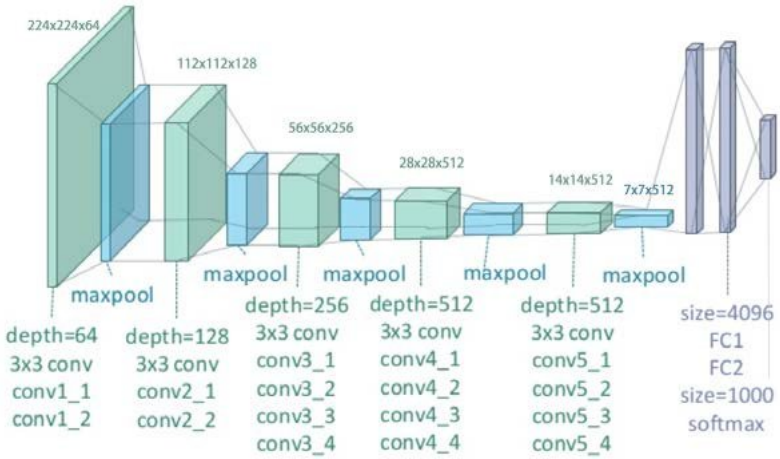


Figure 1: The VGG19 Convolutional Neural Network. 19 layers distribuited in 5 levels ended always with max pooling and unitary stride for a total number of parameters of 144 millions and 3 fully connected layers to achieve the classification task.

3 Methods

In this section we introduce the methods used in the original paper [3] and the mathematical formulas used during the training step. The paper, to obtain a stylized photo, mixes the content representation of an image, that they call *content image* with the style representation of another one, the *style image*. As seen in section 2, the content of an image, despite the exact pixel values, is represented by higher layers' reconstructions, so we expect that the highest level recostruction, the fifth's level feature map ($conv5_1$), is the better choice to select a content representation to mix with the style representation. The style representation is instead obtained by computing the correlations between the different features in different layers of the CNN [3], so we expect that, taking more layers to correlate between them will abstract better the style of an image despite of the content, because this correlation returns only a texturised versions of the input image that capture its general appearance in terms of colour and localised structures that, matched on an increasingly large scale, leads to a smoother and more continuous visual experience [3].

3.1 VGG's truncation

As intuited from the section before we make use of the VGG network to obtain the modified image that is a mix of two pre-defined pictures referred as *Content image*, and *Style image*. However to do so we don't need the full VGG network but only of some of its layers. This model is composed by a number of convolutional plus pooling layers and a final couples of fully connected layers, but what is really needed is only the first ones, so we can truncate the network in order to obtain a shorter model used during the training. The reason of this truncation is that, to confront either the content or the style between two images we don't need the last fully-connected layers that have the only purpose to take the end result of the convolution/pooling process and to reach a classification decision, but instead we need the convolutional outputs given by a convolutional process that makes use of the "feature learning" [1] and that have the capacity to extract local properties [2] from an input.

3.2 Losses

We define two main losses regarding the *Content* concept and the *Style* one. The convolutional layers are made of a number N_l of filters or feature maps that have the capacity to encode the input image and to abstract concepts, like the boundary of a face or the shoulders of a person. By stacking layers of convolutions on top of each other, we can get more abstract and in-depth information. As the original paper states [3], indeed, from the first convolutional layers the reconstruction of an image almost match perfectly the real image, while the last ones don't, but, as already mentioned in the previous sections, they have a significant content rapresentation. Let be $\vec{p}$ and $\vec{x}$ be the original image and the image that is generated, and P^l and F^l their respective feature representation in layer l. We then define the squared-error loss between the two feature representations:

$$L_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} (P^l_{i,j} - F^l_{i,j})^2$$

where M_l is the size of each feature map at the layer l and $F^l, P^l \in \mathrm{R}^{N_l X M_l}$, then $F^l_{i,j}$ is the activation of the i^{th} filter at position j in layer l. Now we define a *Style Loss*. Similar to content loss, Style Loss is used to check how much the style of the generated image differs from the style of the style image. But the difference between them is that the style of any image is not simply represented as in the case of content but instead is given by the correlation of the feature maps. Given two vectors a and b, the correlation between them is given by their dot product:

$$a \cdot b = \sum_i a_i b_i$$

In our case, we want to define the correlation among all the feature maps inside a layer l, so for each couple of feature maps inside the layer l we define the correlation between the i^{th} feature map and the j^{th} feature map obtained vectorizing them (or flatting) and computing their dot product:

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

For this reason the paper makes use of the Gram Matrix $G^l \in \mathrm{R}^{N_l X N_l}$ to compute the style loss for each style layers: given an image mapping, the element at the i^{th} row and j^{th} column of the Gramian matrix would contain the dot product between the flattened i^{th} layer and the flattened j^{th} layer. Moreover by including the feature correlations of multiple layers, we obtain a stationary, multi-scale representation of the input image, which captures its texture information but not the global arrangement. The *Style loss* is then computed using the mean squared distance, in this case between the gram matrices of the style image and the image generated. Let $\vec{a}$ and $\vec{x}$ be the original image and the image that is generated, and A^l and G^l their respective style representation in layer l. The contribution of layer l to the total loss is:

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

and the Style loss is finally computed as follows:

$$L_{style} = \sum_l^L w_l E_l$$

Where L are the number of all the output convolutional layers that have been considered for the loss computation, and w_l is a weightening factor for each layer, chosen arbitrarily. Finally the loss is then computed as:

$$L_{total} = \alpha L_{content} + \beta L_{style}$$

where also in this case α and β are weightening factors.

3.3 Total Variation Loss

For what concern our work we decided to include a further loss that acts as a sort of regularizer in the training of the image. The purpose of this loss defined as $Total\ Variation\ Loss$ [6] is to ensure spatial continuity and smoothness in the generated image to avoid noisy and overly pixelated results. The loss is computed by finding the difference among neighbour pixels, and it helps in denoising the final image. It has been shown to be effective in a number of applications [7] - [8] and is defined as the sum of the absolute differences for neighboring pixel-values in the image. For a 2D image it can be computed according to the following formula:

$$L_{variation} = \sum_{i,j} |y_{i+1,j} - y_{i,j}| + |y_{i,j+1} - y_{i,j}|$$

In this way we finally define the new Total loss as:

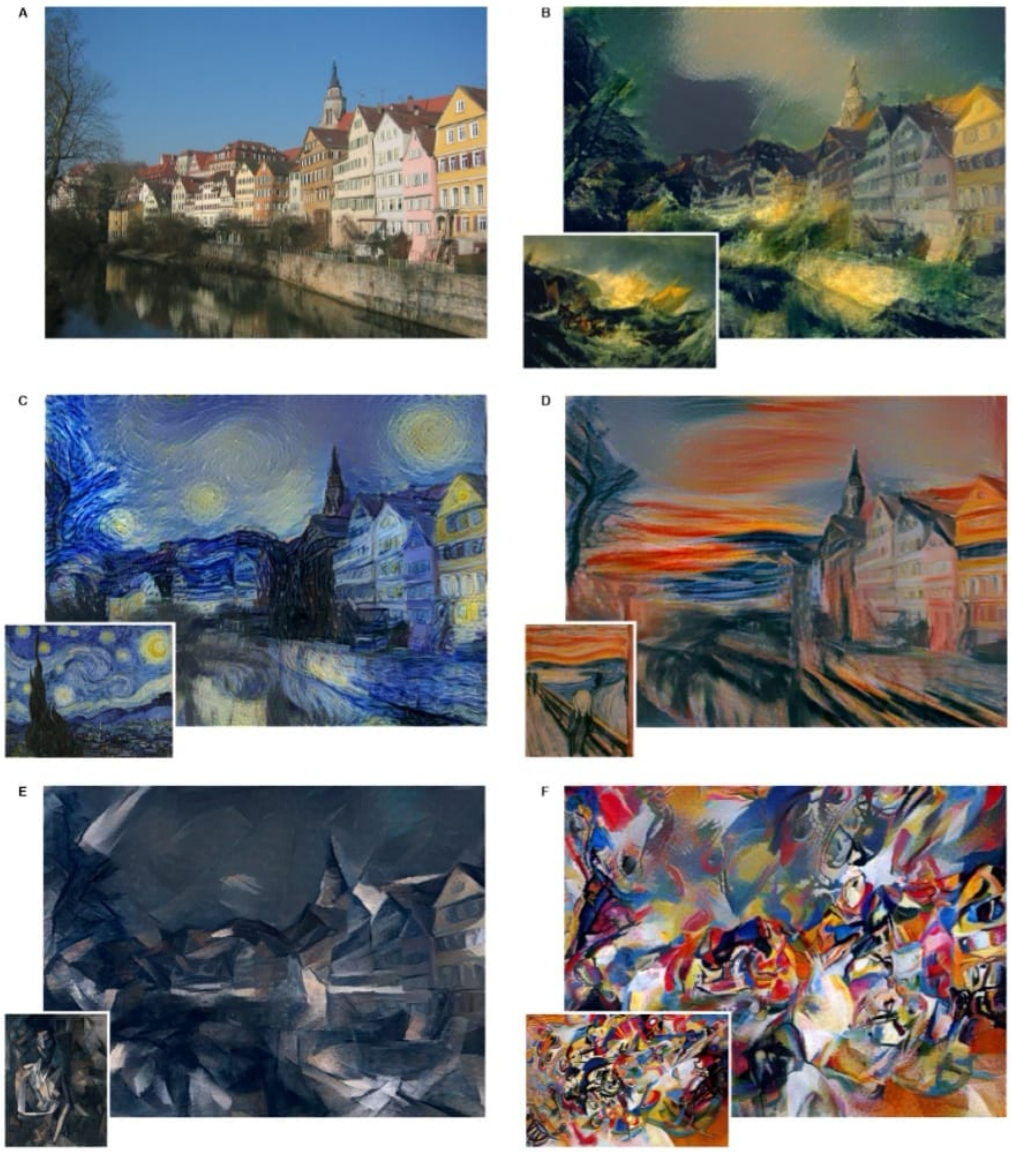
$$L_{new\_total} = \alpha L_{content} + \beta L_{style} + \lambda L_{variation}$$

Where in this case λ acts in turn as a weightening factor for the denoising loss. Adding the total variation loss to the training loss removes the rough texture of the image and the resultant image looks much smoother.

4 Our Work

We re-implemented the work of the paper, where the authors discovered that the representations of content and style in the Convolutional Neural Network are separable, and to enphatise this discover they manipulate both style and content representation mixing them to obtain new perceptually meaningful images [3]. In particular, we match the content representation of a photograph depicting the "Neckarfront" in Tubingen, Germany and the style representations of several well-known artworks taken from different periods of art (Subsection 4.1). A strong emphasis on style will result in images that match the appearance of the artwork, effectively giving a texturised version of it, but hardly show any of the photograph's content (Subsection 2: Fig 3 paper, first column). In the other hand if we place strong emphasis on content the photograph can be clearly identified but the style of the painting is not well-matched (Subsection 2: Fig 3 paper, last column) [3]. Also the alpha/beta ratio (see Methods) must be adjusted for a specific pair of source images to obtain the right trade-off between content and style and to create visually more appealing images. Then we make different experiments variyng the hyperparameters (such as the learning rate, the alpha/beta ratio, using different style or/and content layers, etc) to see which ones show a better result. We implemented our model on the Google Colab platform, a cloud platform that offers the computational power and the space needed for high computing task, like neural network's training steps etc, that otherwise would require lot of time in the low-medium quality range Computers. For the management of the VGG19 network and for building the model that is able to mix the content of an image with the style of another one we used the Tensorflow [1] library. We also used numpy[2] and matplotlib[3]: the first to convert the tensor of the network into arrays and the last for plotting purposes. Then we imported other libraries to download the content or the style image from a given link and to save both in the internal

storage, together with the final result. To evaluate the performance of the model the best metric is the aspect that the result has, so our work was to change the hyperparameters of our network in order to have a good compromise between the content and the style and to have a result that is meaningful, even if in this particular scenario the more appealing result is also subjective. To give a bit more of objectivity in our results and to analize them in order to see if the model had the right hyperparameters, we referred to the total loss (see Methods) between the result and both the content and the style layers. We provided each experiment with the loss plot over the training epochs. We discarded the first two epochs in the representation because they have always too high and meaningless loss, due to the fact that the model is applying gradient descent to a white noise image, so it takes more steps to converge to the mixed result than using the content image also as training image, but we found that it returns better result, of course with more training steps. Finally we can easily see, while executing the code of each experiment in Colab, that the average time to perform 100 iterations of the gradient descent is 7 seconds, with about 8 GB of constant usage of the GPU and 3 GB of the RAM that the cloud platflorm offers. We repeated all the experiments made in the paper [3], all with the same content image, the Neckarfront in Tubingen Germany, in particular the paper made some experiments with the following settings: Figure 2: combines the content image with several well-known artworks. A: no artwork, B: The Shipwreck of the Minotaur by Turner. C: The Starry Night by Vincent van Gogh. D: Der Schrei by Edvard Munch. E: Femme nue assise by Pablo Picasso. F: Composition VII by Wassily Kandinsky. The paper's figure 3: the mixing of the content image with a fixed style image: Composition VII by Wassily Kandinsky, varying along the columns the ratio between the content and the style weight, from 10^{-5} to 10^{-2}, and along the rows different style layers: A: $conv11$ B: $conv11$ and $conv21$ C: $conv11, conv21$ and $conv31$ D: $conv11, conv21, conv31$ and $conv41$ E: $conv11, conv21, conv31, conv41$ and $conv51$.

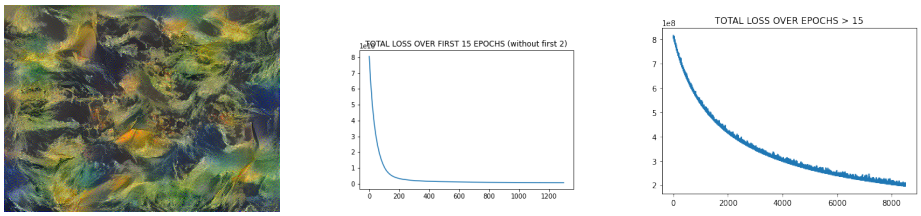

A
B
C
D
E
F

The parameters that are explicitly expressed in the paper are used as they appear in [3], otherwise chosen to have better results.

The shipwreck of the minotaur

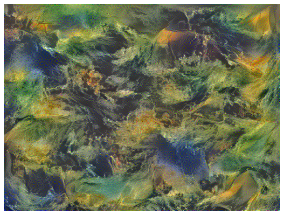
As suggested by the paper, the α / β ratio is 10^{-3}. The other parameters suggested are: Style layers: $conv1_1, conv2_1, conv3_1, conv4_1, conv5_1$. Content layers: $conv4_2$. Total variation weight: 0. The parameters not explicitly expressed are: Optimizer: ADAM, learning rate: 0.001, 0.1, 0.005. Num epochs: 100, steps per epoch: 100, for a total of 10000 iterations of the gradient descent algorithm to reduce the total loss (see Methods). We noted that, with the paper's parameters, in particular using the α / β ratio suggested, the mixed image tended to be confused and meaningless. In fact, even with 10000 iterations (100 epochs),

the total loss tended to stabilize around 10^8 in all our attempt with different learning rate (fig. 4, 5, 6 c). The reason is that with a too high weight on the style the model isn't able to return a result that reminds the content of an image (in this case the buildings) because the training procedure (see Methods) tends to transform the white noise image ever more similar to the style image. Our results are shown in figure 4,5 and 6. In particular fig. 4 a) shows the mixed image with learning rate 0.001 together with the plot of the total loss until the first 14 epochs b) and from the 15^{th} epoch to the last (100) c). Fig 5 a) shows the mixed image with learning rate 0.005 together with the plot of the total loss until the first 14 epochs b) and from the 15^{th} epoch to the last (100) c). Fig 6 a) shows the mixed image with learning rate 0.01 together with the plot of the total loss until the first 14 epochs b) and from the 15^{th} epoch to the last (100) c). We can see that, in all the three attempts, the final mixed image is meaningless for the aspect (the buildings are not visible), and all the final losses are very high (around 10^8). In particular with a greater learning rate (0.01, fig. 6 c)) the loss tends to stabilize earlier to 4 x 10^8. In the other hand, a learning rate lower (0.001, fig 4 c)) lets the loss to reduce more but, even after 100 epochs, doesn't reach values lower than 10^8. Finally, as we can clearly see in all figures 4,5,6 b) , the total loss rapidly decrease in the first 1400 iterations, corresponding to the first 14 epochs.

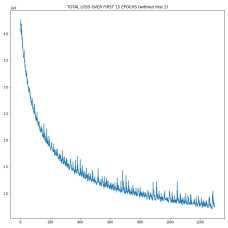


(a) Final image (b) Epochs < 15 (c) $15 \leq Epochs \leq 100$

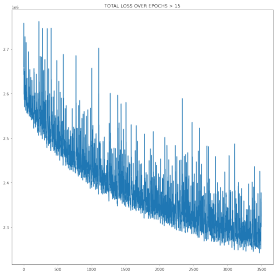
Figure 4: Showing the result using the same parameters of the paper and with learning rate 0.001 a), providing the plot of the loss over the 100 epochs b) and c).



(a) Final image

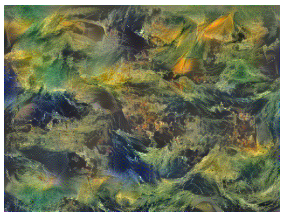


(b) Epochs < 15

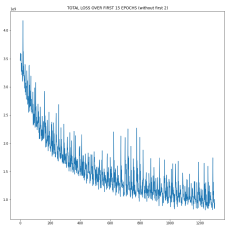


(c) $15 \leq Epochs \leq 100$

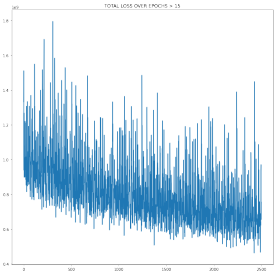
Figure 5: Showing the result using the same parameters of the paper and with learning rate 0.005 a), providing the plot of the loss over the 100 epochs b) and c).



(a) Final image



(b) Epochs < 15

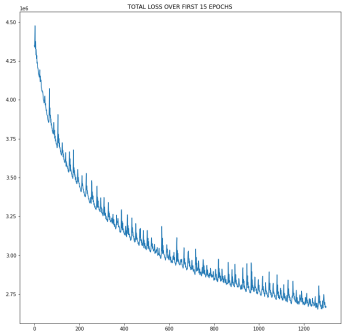


(c) $15 \leq Epochs \leq 100$

Figure 6: Showing the result using the same parameters of the paper and with learning rate 0.01 a), providing the plot of the loss over the 100 epochs b) and c).

In figure 7 we show the results with our parameters that in our opinion perform a better result and also similar to the paper's one, equipped with the loss plot. The main change that gets us to this improvement is due to the re-weight of the content against the style: in particular our α / β ratio is 10^6. We also used the total variation loss introduced in Methods, set its weightening factor to 50.. We trained our model with a learing rate of 0.01 for a total of 5000 steps (50 epochs). The other parameters are the same of all experiments in this subsection.

(a) Final image



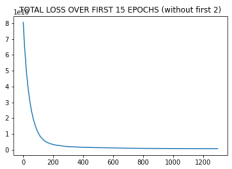
(b) Epochs < 15

Figure 7: The more meaningful result with our hyperparameters a), with the respective total loss.

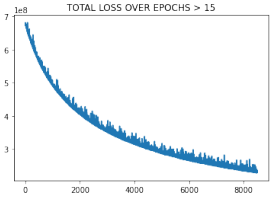
Starry Night

As suggested by the paper, the α / β ratio is 10^{-3}. The other parameters suggested are: Style layers: $conv1_1, conv2_1, conv3_1, conv4_1, conv5_1$. Content layers: $conv4_2$. Total variation weight: 0. The parameters not explicitly expressed are: Optimizer: ADAM, learning rate: 0.001. Num epochs: 100, steps per epoch: 100, for a total of 10000 iterations. As in the previous subsection, the result is meaningless with the paper's α / β ratio and the total loss stabilizes around 10^8 in all our attempt with different learning rate (fig. 8 b)). The results are shown in figure 8, in particular fig.8 a) is our mixed result, fig.8 b) and c) are respectively the loss plot over the first 15 epochs (without the first 2 for the same reason of the previous subs.) and for the others, until the 100^{th} epoch. As before, we found that even changing the other not explicitly defined hyperparameters of the paper the result is still meaningless and the loss stabilizes on the same values. Finally we found that setting the α / β ratio to 3 x 10^6 and the total variation still to 0 the result appear a lot more like the artistic mix that the paper shows. The other hyperparameters are the same of the Figure 7, except for the number of epochs, that in this case is 30, because even if the loss isn't stabilized yet, the more appealing results (visually) are between the 25^{th} and the 35^{th} epoch. We show our result in fig. 9.

(a) Final image

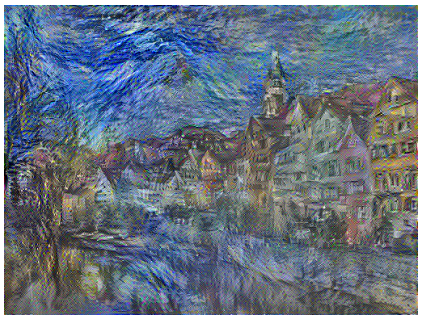


(b) Epochs < 15

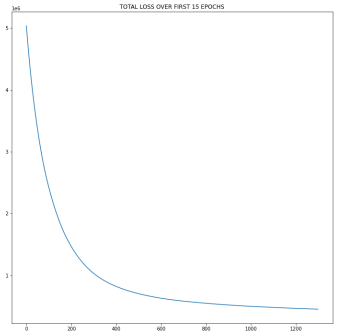


(c) $15 \leq Epochs \leq 100$

Figure 8: The result with the paper's hyperparameters a) and the total loss plot over the 100 epochs b) and c).



(a) Final image



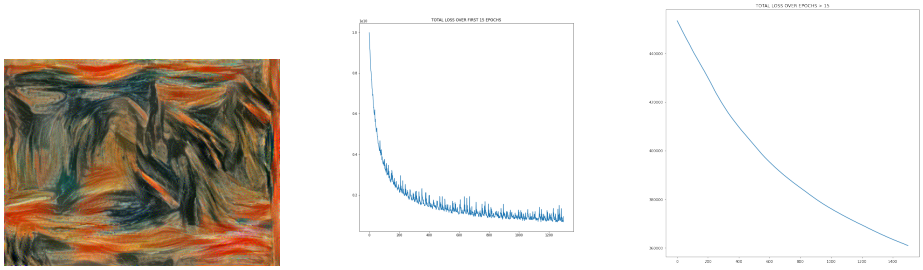
(b) Epochs < 15

Figure 9: The more meaningful result with our hyperparameters a), with the respective total loss.

The Scream

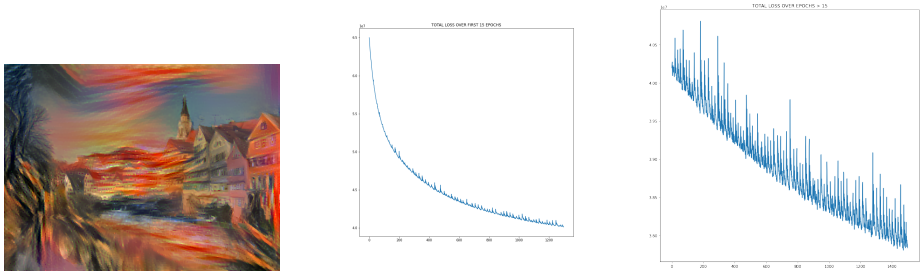
As suggested by the paper, the α / β ratio is 10^{-3}. The other parameters suggested are: Style layers: $conv1_1, conv2_1, conv3_1, conv4_1, conv5_1$. Content layers: $conv4_2$. Total variation weight: 0. The parameters not explicitly expressed, so we choose the better ones are: Optimizer: ADAM, learning rate: 0.001. Num epochs: 100, steps per epoch: 100, for a total of 10000 iterations. the result is meaningless with the paper's α / β ratio and the total loss stabilizes around 10^8 in all our attempt with different learning rate (fig. 10 b)).
The results are shown in figure 10, in particular fig.10 a) is our mixed result, fig.10 b) and c) are respectively the loss plot over the first 15 epochs and for the others, until the 100^{th} epoch. As before, we found that even changing the other not explicitly defined hyperparameters of the paper the result is still meaningless and the loss stabilizes on the same values. Finally we found that setting the α / β ratio to 3 x 10^3 and the total variation to 500 the result appear a lot more like the artistic mix that the paper shows fig.11. The other hyperparameters are

the same of the Figure 10, except for the learning rate and the epochs, that in this case are 0,005 , instad of 0,001 of the figure 10 one, and 30 epochs. As before we found that, even if the loss continues to go down after the epoch 30 (fig.11 c)), the more appealing result is given at that epoch.



(a) Final image (b) Epochs < 15 (c) $15 \leq Epochs \leq 100$

Figure 10: The result with the paper's hyperparameters a), and the plot loss over the 100 epochs b) and c).



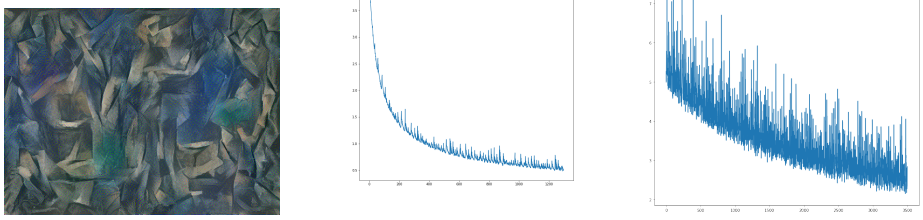
(a) Final image (b) Epochs < 15 (c) $15 \leq Epochs \leq 100$

Figure 11: The result with our hyperparameters a), and the loss b) and c) over 30 epochs.

Femme Nue

As suggested by the paper, the α / β ratio is 10^{-4}. The other parameters suggested are: Style layers: $conv1_1, conv2_1, conv3_1, conv4_1, conv5_1$. Content layers: $conv4_2$. Total variation weight: 0. The parameters not explicitly expressed, so we choose the better ones (after trying

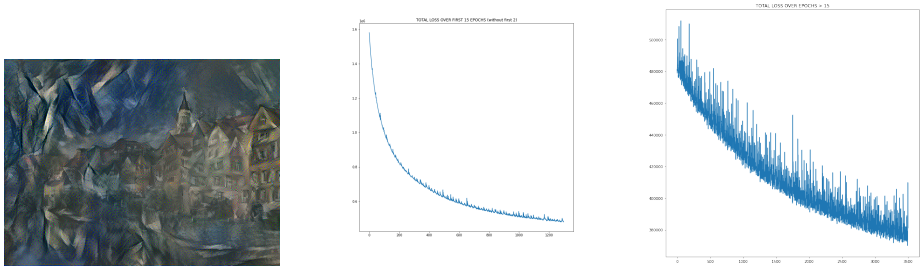
all possible combinations) are: Optimizer: ADAM, learning rate: 0.005. Num epochs: 50, steps per epoch: 100, for a total of 5000 iterations. The results are shown in figure 12, in particular fig.12 a) is our mixed result, fig.12 b) and c) are respectively the loss plot over the first 15 epochs and for the others, until the 50^{th} epoch.



(a) Final image (b) Epochs < 15 (c) $15 \leq Epochs \leq 100$

Figure 12: Our result following the paper's hyperparameters a) with the loss over the 50 epoch b) and c).

Figure 13 shows our hyperparameters' result with: Style layers: $conv1_1$, $conv2_1$, $conv3_1$, $conv4_1$, $conv5_1$; Content layers: $conv5_2$; α / β ratio: 10^5; Total variation weight: 0; optimizer Adam with learning rate: 0,005 and 50 epochs. Also in this case we found that was better to stop at the 50^{th} epoch, even if the loss was still going down (figure 13 c)).

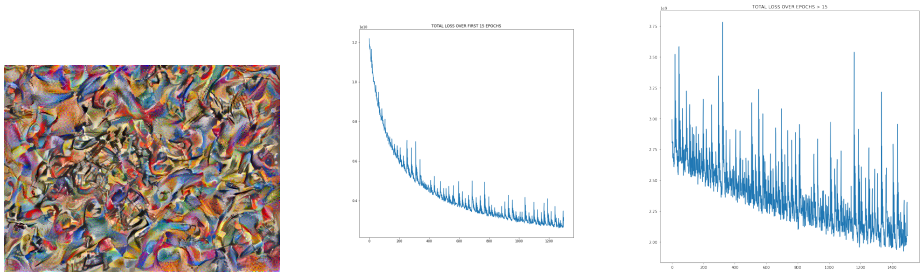


(a) Final image (b) Epochs < 15 (c) $15 \leq Epochs \leq 100$

Figure 13: The result with our hyperparameters a) and the loss over the 50 epochs b) and c).

Composition VII

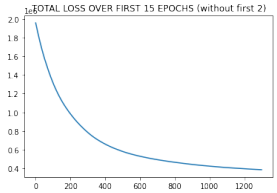
As suggested by the paper, the α / β ratio is 10^{-4}. The other parameters suggested are: Style layers: $conv1_1, conv2_1, conv3_1, conv4_1, conv5_1$. Content layers: $conv4_2$. Total variation weight: 0. The parameters not explicitly expressed, so we choose the better ones (after trying all possible combinations) are: Optimizer: ADAM, learning rate: 0.005. Num epochs: 50, steps per epoch: 100, for a total of 5000 iterations. The results are shown in figure 14, in particular fig.14 a) is our mixed result, fig.14 b) and c) are respectively the loss plot over the first 15 epochs and for the others, until the 50^{th} epoch. Figure 15 shows our hyperparameters' result with: Style layers: $conv1_1, conv2_1, conv3_1, conv4_1, conv5_1$. Content layers: $conv5_2$. α / β ratio: 10^5; Total variation weight: 0; optimizer Adam with learning rate: 0,005 and 50 epochs. Also in this case we found that was better to stop at the 50^{th} epoch, even if the loss was still going down (figure 15 c)).



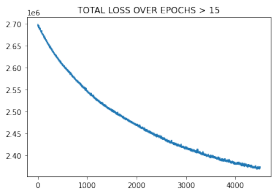
(a) Final image (b) Epochs < 15 (c) $15 \leq Epochs \leq 100$

Figure 14: The result with paper's hyperparameters

(a) Final image



(b) Epochs < 15



(c) $15 \leq Epochs \leq 100$

Figure 15: The result with our hyperparameters

5 Conclusion

In this paper we have shown the settings needed to re implement the paper "A Neural algorithm of Artistic style", in particular the different hyperparameter settings to obtain the best result from the Vgg pretrained network. In addition we have also shown how the regulizer factor regarding the total variation loss could play an important role during the training phase from a white noise image. Nothing excludes in future different settings that achieve an even better performance.

Bibliography

[1] Yoshua Bengio, "**Deep Learning of Representations for Unsupervised and Transfer Learning**",(JMLR Workshop and Conference Proceedings),2012.

[2] Yoshua Bengio, Lecun, Yann "**Convolutional Networks for Images, Speech, and Time-Series**",(JOUR),1977.

[3] Leon A. Gatys, Alexander S. Ecker, Matthias Bethge "**A Neural Algorithm of Artistic Style**",2015.

[4] Karen Simonyan, Andrew Zisserman, University of Oxford "**Very Deep Convolutional Networks for Large-Scale Image Recognition** ",2015.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, Li Fei-Fei "**Imagenet: A Large-Scale hierarchical Image Database** ",(IEEE conference on computer vision and pattern recognition), 2009.

[6] Jiaming Liu, Yu Sun, Xiaojian Xu, Ulugbek S. Kamilov "**Image Restoration using Total Variation Regularized Deep Image Prior** ",(arXiv), 2018.

[7] M. Persson, D. Bone, and H. Elmqvist "**"Total variation norm for three-dimensional iterative reconstruction in limited view angle tomography** ",(Phys. Med. Biol., vol. 46, no. 3, pp. 853–866), 2001.

[8] U. S. Kamilov, I. N. Papadopoulos, M. H. Shoreh, A. Goy, C. Vonesch, M. Unser, and D. Psaltis "**Optical tomographic image reconstruction based on beam propagation and sparse regularization** ",(IEEE Transactions on Computational Imaging, vol. 2, no. 1, pp. 59–70), 2016.