Artistic Style Transfer Using VGG19

1st Tomás Oelckers A.
Facultad de Ingeniería y Ciencias Aplicadas
Universidad de los Andes
Santiago, Chile
toelckers@miuandes.cl

2nd Tarik S. Riadi
Facultad de Ingeniería y Ciencias Aplicadas
Universidad de los Andes
Santiago, Chile
tsaez@miuandes.cl

***Abstract*—This paper presents a thorough analysis of artistic style transfer using a VGG19 convolutional neural network, to measure its execution time, style and content losses and qualitative performance of the final result.**

***Index Terms*—style, transfer, image processing, convolutional, neural, network, artificial, intelligence**

I. Introduction

The most promising Deep Neural Network architecture in image processing tasks are Convolutional Neural Networks (CNNs). Convolutional Neural Networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers [1]. CNNs are very useful for extracting information about an image from its layers, information that can be present in the image in no trivial forms of traditional human understanding, and that is why this type of NN are so useful in artistic style transfer.

Style transfer, simply put, is to emulate the painting style (strokes, colors, tones, among other features) of a pastiche and implement it to another image without deforming it completely, such that content of the original image is preserved while achieving the artistic style of the pastiche. In order to do this, one must successfully extract the style, along with its underlying features, while avoiding extracting content from the painting (for example a person, a house or other objects that could be present in the pastiche). This is why CNNs provide the necessary tools for extraction: because they have been trained to find features within an image.

This paper shows the results of the implementation of artistic style transfer using a pre-trained CNN: VGG19 [2], based on the research of Gatys *et al.* [3]. The code used to do so is present in a Jupyter Notebook Colaboratory [4].

II. VGG19

The VGG19 NN consists of 19 convolutional network layers; out of the six chosen layers, five of them are intended to capture the pastiche's style and just one is destined to capture its content. These layers were chosen in an iterative process of trial and error to find the most accurate result. It is important to point out that a pre-trained version of VGG19 with ImageNet database was used.

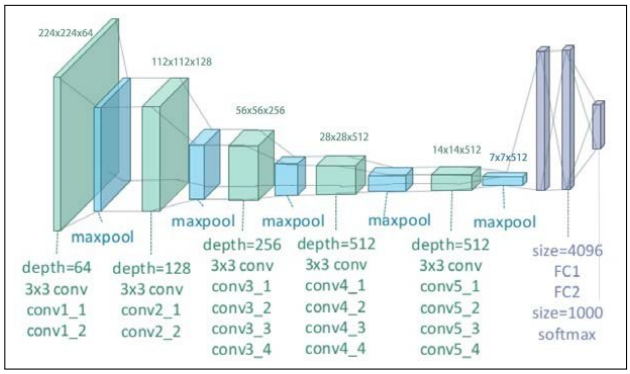


Fig. 1: VGG19 architecture diagram. "conv" means convolution and "FC" means fully connected. These model's parameters are not the same used for the style transfer, but it is based on the same architecture. Only dimensions and number of layers are different. [2]

TABLE I: VGG19 layers corresponding to the style and content capture from the painting. Out of the 21 layers in the NN, these are the ones that "absorb" style and content from the image.

Layer Name	Function
block1_conv1	Style
block2_conv1	Style
block3_conv1	Style
block4_conv1	Style
block5_conv1	Style
block5_conv2	Content

III. Image Style Transfer

The algorithm implemented to perform the artistic style transfer was fundamentally based in already existing code [5], where some modifications to the parameters and order of the implementation were made[1]. To better understand what is going on in style transfer, one must first define the two images involved in each process: The original image and the pastiche. The original one makes reference to the image to which the

[1]As presented in the "Deliverable 2: Artistic Style Transfer Using VGG19" work.

artistic style transfer will be applied onto; the pastiche is the selected artist's painting, from which the style will be extracted to be applied to the original one.

A. Style Transfer Function

This function makes the actual style transfer from the pastiche onto the original image through the feature extraction made in the intermediate layers of the VGG19 NN defined in I. It is important to highlight that the NN used is pretrained and the weights of each layer are not modified; in other words, the NN is not retrained during the style transfer process.

B. Loss Function

It is important to account for the losses that happen during the content and style extraction from each of the two images. These extractions rely heavily on the weights defined for content and style explained in the previous section; nevertheless, it is impossible to avoid such losses during the style transfer. The following equation allow to quantify them and measure these losses, which enable one to not only compensate theses losses to enhance the final result, but also to evaluate the method itself.

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

Much of the losses happen to avoid transferring high-frequency components from the pastiche onto the original image, to produce a more aesthetically pleasant and credible result.

C. Optimization Function

There are several optimization algorithms available for *tensorflow* and *keras* within the google colab environments. Out of them all, two stand out the most: Adam's optimization and L-BFGS. There are several studies and tests [6] show that these two get the best aesthetic results in the less amount of time and with a minimum loss of both content and style, compared to other optimization algorithms. The problem of the implementation of L-BFGS is that it requires more functions, and therefore more code, to be implemented; overall, neither its time execution nor its content and style losses outperform Adam's one significantly.

1) Adam's Optimization Learning Rate: In order to obtain the best results in terms of style transfer and execution time, several learning rates values were tested. The setup used was the *google colab* assigned computer with the GPU as its runtime type, and all tests were conducted in the same virtual machine.

IV. Results

To analyze the performance of the implemented style transfer, five tests were made with different learning rates (0.1, 0.2, 1, 5 and 10), 1000 iterations each, and the content and style weights from equation (1), $\alpha = 1000$ and $\beta = 0.01$ respectively, were not modified because these values show better results for this method, making the content prevail over the style.

In figure (5), images (b) to (f) represent artistic style transfer for the learning rate defined before. It becomes evident that higher learning rates transfer style more heavily than lower ones, at the cost of the original image's content. What this means, is that the image, although more true to the artist's style, it loses its original shape; for example, from figures (b) to (d), the eyes of the subject preserve its original form. However, figures (e) and (f) deform it to the point that the eyes are no longer recognizable; furthermore, the objects in the background lose their shape too.

In figures (2), (3) and (4), one can see that as the learning rate increases, the losses converge more rapidly. In addition to this, one can notice that images with higher learning rates acquire more of the pastiche's style and therefore content and style losses are decreased. What this means, is that there are less differences between style and content from the output images of figure (5) and its corresponding pastiches; in other words, they preserve more of the pastiche's style, but also its content. This adds more noise and overall deformation to the original image, but also seem more "artistic".

It is remarkable that learning rate does not affect execution time; however, the style transfer changed significantly between learning rates. Lower learning rates preserved the original image's content more, but needed more learning iterations to reach the same aesthetic results that higher learning rates obtained. For example the result of a learning rate of 1 at iteration 1000 was obtained by a learning rate of 5 at iteration 300. From this, one can conclude that execution time can be optimized by running less iterations of higher learning rates, as the running time for all learning rates was practically the same.

TABLE II: Learning rate vs execution time of the style transfer using VGG19 and Adam's optimization algorithm.

Learning Rates	Execution Time
0.1	266.41
0.2	242.65
1	242.21
5	243.54
10	243.19

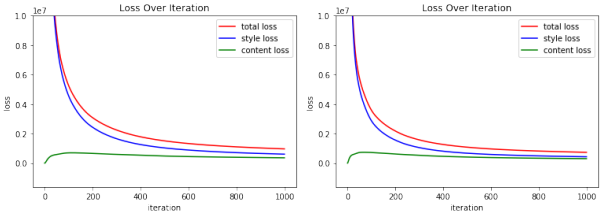
On the other hand, the choice of learning rate and number of iterations is not trivial because a low-learning rate, high-iteration number NN does not achieve the same results of one that ran with a higher learning rate and less iterations,

suggesting that these arguments do not have the same impact on the output image. Furthermore, fewer iterations[2] (despite of the learning rate) can actually negatively impact the output's result because there will be more high-frequency artifacts and a less natural color distribution throughout the "painting".

It became evident after several tests that higher learning rates transferred more content from the pastiche onto the original image, and also less iterations failed to dispose high-frequency noise from the outputs; This can be appreciated in Piet Mondrian's style transfers (figure (8)).

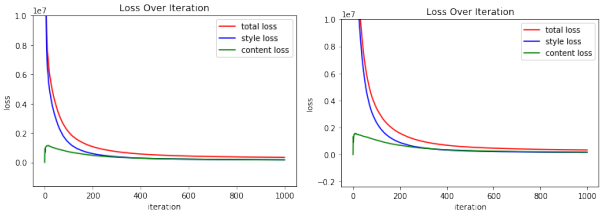
TABLE III: Content, style and total loss for every learning rate after 1000 iterations.

Learning Rates	Content Loss	Style Loss	Total Loss
0.1	357,243.3	605,918.8	963,162.1
0.2	300,354.28	428,854.34	729,208.6
1	212,622.73	235,394.17	448,016.9
5	171,222.31	165,308.06	336,530.38
10	179,215.89	161,174.42	340,390.3



(a) Learning Rate = 0.1 (b) Learning Rate = 0.2

Fig. 2: Loss of content, style and total over every iteration of the NN runtime for learning rates of 0.1 and 0.2.



(a) Learning Rate = 5 (b) Learning Rate = 10

Fig. 3: Loss of content, style and total over every iteration of the NN runtime for learning rates of 5 and 10.

V. Conclusion

The following can be concluded about this work:

1) The image style transfer method from Gatys works properly using the layers (Table I) of CNN VGG19 for content and style extraction.
2) The Adam optimizer shows the best results, in terms of loss and performance, for learning rate values between 1 and 5.

[2]The NN should not be ran with less than 1000 iterations.

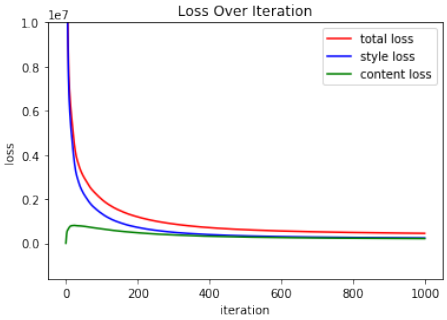


Fig. 4: Loss of content, style and total over every iteration of the NN runtime for learning rate of 1.

3) It was demonstrated that different learning rates do not affect execution runtime.
4) Changing the learning rate only accelerates the search process of the minimum loss between content and style; therefore, using learning rates outside of the boundaries of point 2 can only make the final result worse, qualitatively speaking.
5) Also, to obtain a nice result of style transfer, using the layers selected in table I, it is necessary that the weight of content loss was significantly bigger than the weight of style loss.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, "Convolutional networks," in *Deep Learning*, MIT Press, 2015, ch. 9, p. 330.

[2] Y. Zheng, C. Yang, and A. Merkulov, "Breast cancer screening using convolutional neural network and follow-up digital mammography," May 2018, p. 4. DOI: 10.1117/12.2304564.

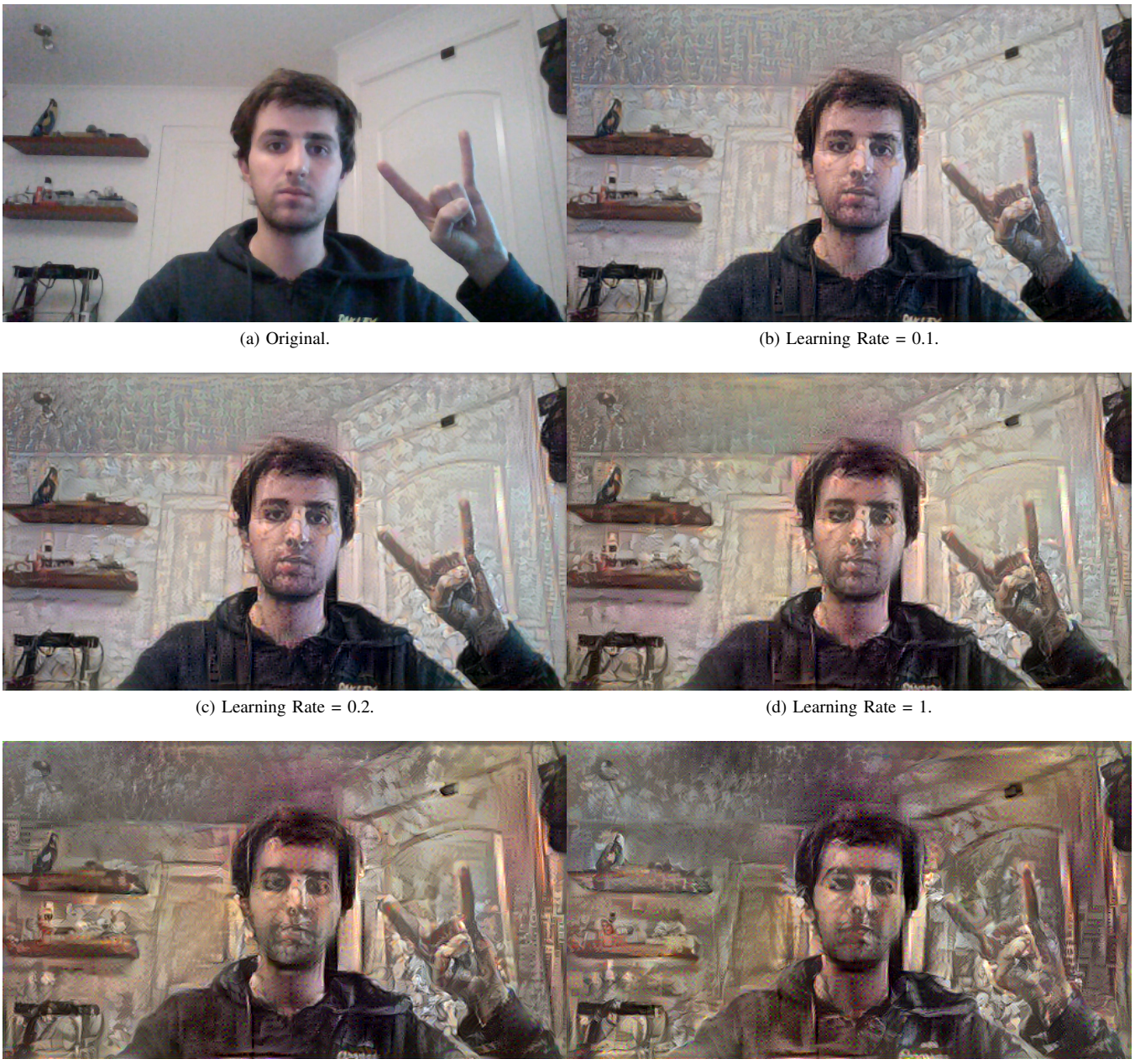
[3] L. A. Gatys, A. S. Ecker, and B. Mathias, "Image style transfer using convolutional neural networks," *IEEE Xplore*, pp. 2414–2423, Dec. 2016.

[4] T. Oelckers and T. S. Riadi. (Nov. 2019). Artistic Style Transfer Using VGG19, [Online]. Available: https://colab.research.google.com/drive/109suyeFK6baV-vlsIRid3-MYX7Me_XAe.

[5] Tylersuard. (Oct. 2019). Neural Style Transfer with Eager Execution, [Online]. Available: https://colab.research.google.com/github/tensorflow/models/blob/master/research/nst_blogpost/4_Neural_Style_Transfer_with_Eager_Execution.ipynb#scrollTo=jo5PziEC4hWs.

[6] S. Ivanov. (Mar. 2017), [Online]. Available: https://blog.slavv.com/picking-an-optimizer-for-style-transfer-86e7b8cba84b.

Appendix



(a) Original.

(b) Learning Rate = 0.1.

(c) Learning Rate = 0.2.

(d) Learning Rate = 1.

(e) Learning Rate = 5.

(f) Learning Rate = 10.

Fig. 5: Output images using Diego Velazquez's "Las Meninas" (figure (6, a)) for different learning rates, each after 1000 iterations.

(a) Diego Velasquez's "Las Meninas"(b) Piet Mondrian's Blue Flowers. zoomed-in.

(c) Alfred Sisley's Snowy Landscape.

Fig. 6: Pastiches used for style transfer.

(a) Original landscape.

(b) Learning Rate 4, $\alpha = 100$, $\beta = 0.1$.

(c) Learning Rate 4, $\alpha = 1000$, $\beta = 0.01$.

Fig. 7: Output images using Alfred Sisley's pastiche of a snowy landscape. Figure (b) used different content and style weights than the other images. Notice how the original image's content is better preserved in (c) than (b) because of the selected weights.

(a) Learning rate = 1

(b) Learning rate = 5

Fig. 8: Output images using Piet Mondrian's painting (figure (6, b)) for style transfer after 1000 iterations.