Neural Style Transfer
Implementation and analysis of various techniques

Mattia Lisciandrello
Polytechnic University of Turin
s286329@studenti.polito.it

Gabriele Inzerillo
Polytechnic University of Turin
s287028@studenti.polito.it

Antonino Monti
Polytechnic University of Turin
s287037@studenti.polito.it

Abstract

Deep Neural Networks can be used in a plethora of ways, and they can be applied for various tasks: they can even create artistic images of high quality, as shown in the seminal work of Gatys et al.. We mainly took inspiration from their work, trying to separate and recombine a content image and a style one by following their algorithm. Then, we tried to experiment and expand their work by using another model in order to compare the obtained results. The paper concludes with a discussion of the results obtained.

1. Introduction

Painting has always been one of the most popular forms of art, and has been used for thousands of years by artists as a way to express themselves. Many painters have tried to create new styles, by re-drawing images in new ways. However, this was a time-consuming task. As of today, Deep Neural Networks are being used more and more, and Convolutional Neural Networks represent one of the most powerful tools in the world of image processing. Gatys et al. [1] first studied how to use Convolutional Neural Networks to reproduce famous painting styles, by applying a technique called Neural Style Transfer (NST).

There are many ways and algorithms to transfer the style from an image to another; however, we will focus only on the seminal work of Gatys et al., where they use Convolutional Neural Networks on input images in order to obtain information about the content of a content image and the style of a style image. The information can be seen through the feature maps contained in each layer of the networks. For this reason, deeper layers of the network are used to capture and recognize the high-level content of a specific content image. However, high-level information is not as useful in order to obtain the style of an image: the style of an input image is in fact seen through lower levels of the network, which are used to capture the texture information of an image. One of the most important concepts of Gatys et. al. is the independence between the two representations, as the information about them can be obtained independently of each other: this makes us able to manipulate both and combine them to create new images. So, we have to reconstruct the content of the the content image, the style of the style image and apply the style upon the content image. The result of this process will be an image that will present both the structure of the content image and the colours and texture of the artwork. We can also decide how much each image will weigh in the final product: we can emphasize for example a certain style if we want our image to match our artwork more than the original photography.

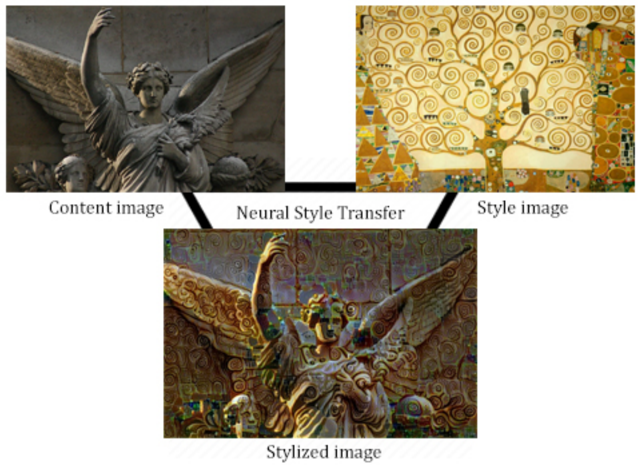

Figure 1. Example of how a Neural Style Transfer works: an image is generated by combining a content image and a style image. The style image is ”The Tree Of Life” by Gustav Klimt.

There are two ways to implement image reconstruction:

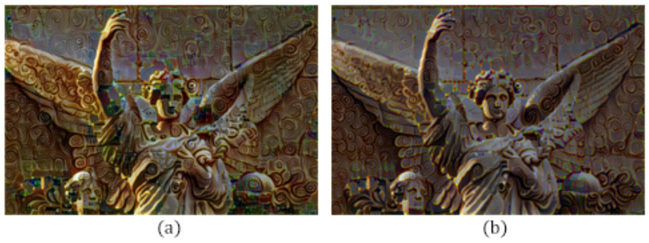

Figure 2. Outputs with different content/style ratios. (a) is the output with a 10^{-4} ratio and (b) with a 10^{-3} ratio.

1) Image-Optimisation-Based, used by Gatys et. al. which is based on the gradient descent in image space. The process however is time-consuming

2) Model-Optimisation-Based, a more efficient approach.

We will focus on the first one since it is the approach used by the work we are trying to replicate; a deeper comparison of the two models can be seen in the work of Jing et. al. [2].

2. Data

As mentioned earlier, the algorithm used by Gatys et. al. will be particularly time-consuming since we will focus on the Image-Optimisation-Based methodology. Also, since the focus of the work is to reconstruct images, we do not need a large dataset: only a relatively small amount of images is needed. We have used two different sources as our data:

1) For content images, we chose to use a pre-existing benchmark dataset called NPRgeneral, which is also referenced in the work of Jing et. al.: it is a dataset of twenty images that cover a wide range of characteristics regarding their quality, such as the contrast of the images and the texture. More information can be found in the work by Mould and Rosin [3]. We chose to use the same dataset both to have an easier way comparing our work and to have a good set of high quality images. [NPRGeneral link]

2) For style images, we chose ten artworks of different styles in order to evaluate how the model would perform with different styles. They have been selected manually by ourselves: in order to do so we used the public dataset present on kaggle [Best Artworks of All Time] or other sources like Google.

Other than the sources, we also needed a way to preprocess our data: the images had to be resized and formatted to obtain appropriate tensors in order to work on them properly, otherwise different images with different dimensions would have been too problematic. Furthermore, we also needed to de-process the tensors to convert them back to images.

We did not perform any data augmentation or splitting of the data since we did not deem it useful for the task.

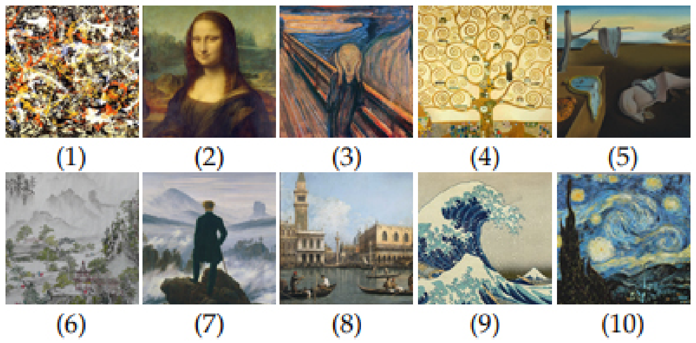

Figure 3. Style images used during the experiments.

Number	Author	Name
1	Jackson Pollock	*Convergence*
2	Leonardo da Vinci	*Monna Lisa*
3	Edvard Munc	*The Scream*
4	Gustav Klimt	*The Tree of Life*
5	Salvador Dalì	*The Persistence of Memory*
6	Chen Minglou	*Part of Giant Traditional Chinese Painting*
7	C. D. Friedrich	*Wanderer above the sea of fog*
8	Canaletto	*Veduta del bacino di San Marco [...]*
9	K. Hokusai	*The Great Wave off Kanagawa*
10	Vincent Van Gogh	*The Starry Night*

Table 1. Informations regarding every artwork.

3. Methods

3.1. Model

In order to perform Neural Style Transfer, we followed the approach explained in the works of Gatys et. al and Jing et. al., and thus we decided to use the pre-trained VGG19 Network [4]. Following the work of the mentioned papers, we used only a part of the network's layers to process the style and content of the respective images.

Particularly, the layers used to process the style image are the following: {*block1_conv1, block2_conv1, block3_conv1, block4_conv1, block5_conv1*}

To process the content image a single layer is used: {*block4_conv2*}. We also tried using the {*block5_conv2*} layer, but no significant difference was found in the results. The original paper by Gatys et. al. also suggests to replace max pooling with average pooling; however, having obtained satisfying results with max pooling we decided not to change those.

We also experimented with two other pre-trained networks: VGG16 and ResNet50. For the first one we used the same layers, while for the second we used the following layers for the style: {*conv2_block3_1_conv, conv3_block1_1_conv,*

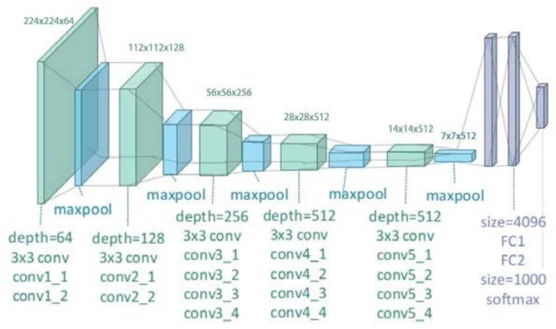

Figure 4. Structure of the VGG19.

conv4_block1_1_conv, conv5_block1_1_conv}, and the following for the content: {*conv5_block1_1_conv*}.

3.2. Loss

One of the most crucial factors in Neural Style Transfer is the computation of losses. In order to train the network to analyze the style of the style image and apply it to the content image, three losses need to be computed: content loss, style loss and total variation loss. The latter was not included in the works of Gatys et. al., and its absence contributed in creating blurry images. Thus we had to resort to this implementation of the total variation loss referenced in the work of Jing et. al.: {titu1994 work} in order to make results more sharp.

Let p and a represent, respectively, the content image and the style image, x represents the final image (stylised image), α represents the content weight, β the style weight. To compute the total loss, the content and style losses are combined as follows:

$$L_{total}(p,a,x) = \alpha L_{content}(p,x) + \beta L_{style}(a,x) \quad (1)$$

with the addition of the total variation loss.

The content and style weights are used to balance the content and the style components. We have tested multiple α/β ratios, but the best results were obtained with a ratio of 10^{-4}.

3.2.1 Gram Matrix and Style Loss

The work of Gatys et. al. defined the style loss as the squared euclidean distance between the Gram-based style representation derived from a style image to that of the styled image, as noted in the work of Jing et. al.. Basically, we compute the distance between the feature map of the two images through the computation of the Gram Matrix for each image. A Gram Matrix represents the inner products between the vectorised feature maps:

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{Jk} \quad (2)$$

Let a be the original image, x the generated image, A, G their respective gram matrix computations, M_l the height times the width of the feature map and n_l the number of distinct filters. The contribution of a specific layer l to the total loss is computed as:

$$E_l = 1/(4N_l^2 M_l^2) \sum_{ij} (G^l_{ij} - A^l_{ij})^2 \quad (3)$$

While the use of this equation contributed in creating satisfying results with the use of VGG19 and VGG16, it did not do the same for the ResNet50 network. During our experiments, we found that ResNet50 did not produce any stylized images, but often merely produced a blurred version of the content image. After several tests, we found that the style loss tended to have very small values, implying that the network was not properly analyzing the style of the style image, thus making the output blurry. For this reason, we found that removing the denominator in equation (3) produced much better results, and thus decided to remove it while working with ResNet50. The total style loss is then computed as such:

$$L_{style}(a,x) = \sum_{l=0}^{L} w_l E_l \quad (4)$$

Where w_l represents the weighting factors of the contribution of each layer to the total loss. The derivative of the loss is then computed using standard back-propagation.

3.2.2 Content Loss

The content loss compares the content representation of a given content image to the one of the generated image.

Let p be the original image, x the generated image, P^l and ^{l} their respective feature representation. The content loss is computed as such:

$$L_{style}(p,x,l) = 1/2 \sum_{i,j} (F^l_{ij} - P^l_{ij})^2 \quad (5)$$

Again, the derivative of this loss is then computed using standard back-propagation.

3.2.3 Total variation Loss

As anticipated, total variation loss is used to make the image smoother and less noisy. Using total variation loss allows to apply a denoising procedure, which encourages pixels close to one another to be more similar in order to generate a more gradual shift in values.

3.3. Hyper-parameters

As mentioned before, we decided to have the α/β ratio be $\approx 10^{-4}$. The total variation loss should be at least 10^{-5}: we have reached the best results with 10^{-6} but it is up to personal preference. We tried a various number of iterations: we settled with 1000, since with more iterations there is not enough difference to justify the higher execution time. As the learning rate, we have tried various learning rates: the default one is 100, with a decay every 100 steps. Depending on the type of picture an higher learning rate was needed.

All these experiments with different hyper-parameters were done in all the 3 models we used: VGG19, VGG16 and ResNet50.

3.4. Most challenging part

One of the most challenging parts was to understand how to update the losses after each iteration. In order to successfully complete the task we have followed various implementations and guides.

Other than that, we have spent several days figuring out both how to make the ResNet work with the functions used for the VGG19 and figuring out the layers to use to extract the features.

3.5. Evaluation

Evalaution in this case is not easy to perform. Metrics can only go so far, since in this situation not only quantitative evaluation is needed (speed, loss), but also qualitative. We mainly focused on the qualitative aspect (trying to make the imagine as good as possible, while reducing noise and making the images more sharp), while still keeping track of the losses and of the speed to evaluate the quantitative performances of the Network.

4. Experiments

We have tried a plethora of experiments in order to understand better the impact of the hyperparameters (like the results with different content/style ratios or different weights for the total variation loss) and the performances of the networks. We also compared different pre-trained models: after tuning the hyperparameters on the VGG19 we tried to use VGG16 and the ResNet50 in order to compare losses and times.

Overall, we found out that the results with VGG19 were more pleasing (but with a notable increase in execution time) [Figure 5] In order to make ResNet50 function properly, we had to update the function used to compute the style loss. Without this, our results were too blurry to even recognize figures. [Figure 6]

During our experiments we tried a variety of content images and style images as said before, in order to understand better how our Neural Style Transfer would perform with different types of pictures [Figure 8].

For example, during these tests we found out that our outputs with portraits needed an higher learning rate to produce good looking results compared to landscapes.

We tried various configurations regarding learning rate, going as low as 10 and as high as 300 and more: this would

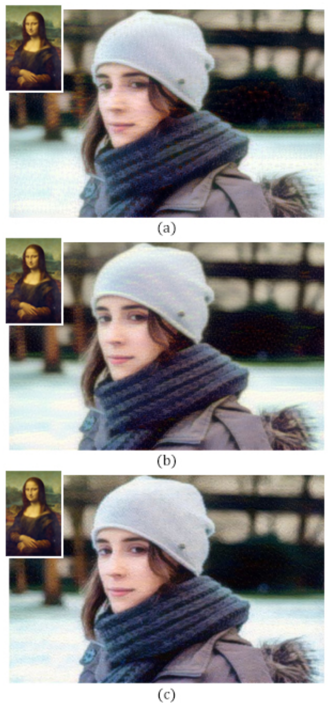

Figure 5. Difference between (a) VGG19 (b) VGG16 (c) ResNet50 with just 1000 iterations.

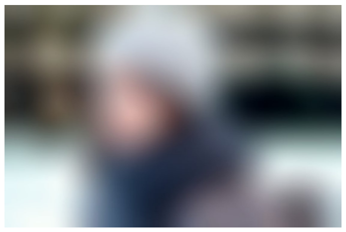

Figure 6. One of the first outputs of our ResNet, before updating the function used to compute the style loss.

Model	Execution time (s)
VGG19	854
VGG16	714
ResNet50	269

Table 2. Different execution times. However, the execution times are not completely reliable since they depend on the service of Google Colab: different GPUs are given at different time. Nonetheless, ResNet50 has always performed better overall.

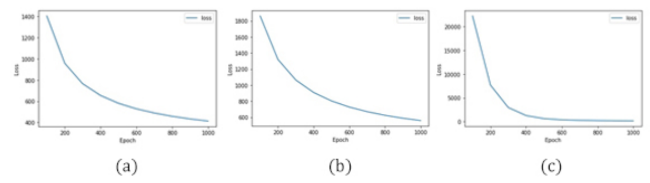

Figure 7. Loss curves comparison. (a) Loss curve with VGG19 (b), with VGG16 and (c) with ResNet50

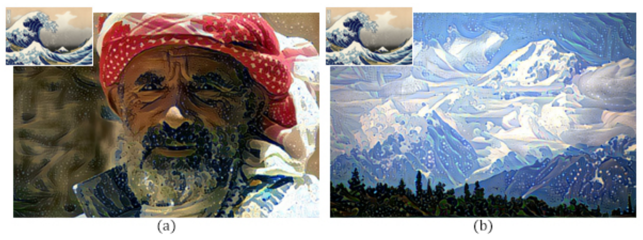

Figure 8. Outputs comparison between portraits and a landscape, both with the same learning rate. (a) Portrait of a yemeni man (b) Landscape of a mountain.

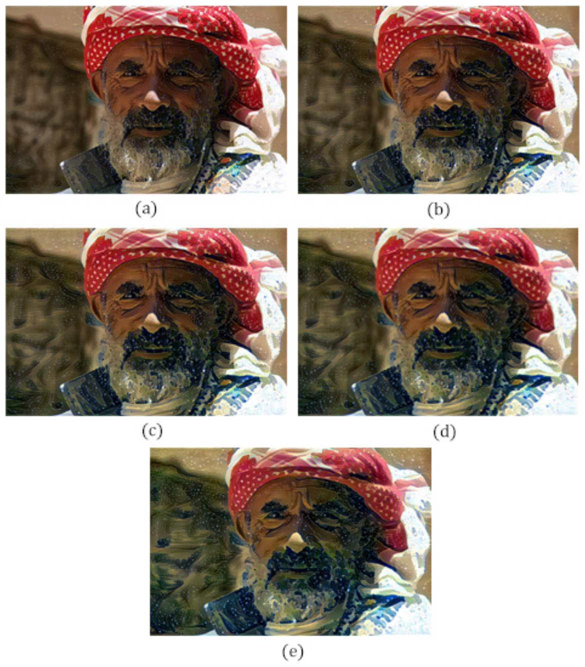

Figure 9. Outputs with different learning rates. (a) Learning rate = 10 (b) Learning rate = 50 (c) Learning rate = 100 (d) Learning rate = 150 (e) Learning rate = 300.

increase the loss value, but in our case it wasn't much relevant since the most important factor is how aesthetically pleasing a certain image is at the end of the process. The value of the loss was dependant on the configuration and on the inputs, but overall we had a strictly decreasing curve nearly every time [Figure 9].

Also we have tried to use different layers of the pre-trained VGG network, but we haven't found any noticeable difference in results [Figure 10].

Figure 10. Outputs with different layers used for the content. (a) Using {*block4_conv2*} (b) Using {*block5_conv2*}. The outputs are basically the same.

We repeated the same experiments with ResNet, by trying multiple configurations both for the content and the style layers [Figure 12].

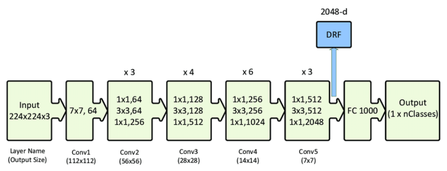

Figure 11. Architecture of the ResNet50.

a)Content{*conv2_block1_2_conv*},
style{*conv2_block3_1_conv,conv3_block1_1_conv,*
conv4_block1_1_conv,conv5_block1_1_conv}
b)Content{*conv2_block1_2_conv*},
style{*conv2_block3_1_conv,conv3_block3_1_conv,*
conv4_block3_1_conv,conv5_block1_3_conv}
c)Content{*conv5_block3_3_conv*},
style{*conv2_block3_1_conv,conv3_block1_1_conv,*
conv4_block1_1_conv,conv5_block1_3_conv}
d)Content{*conv4_block2_2_conv*},
style{*conv2_block3_3_conv,conv3_block3_3_conv,*,
conv4_block3_3_conv, conv5_block3_3_conv}
e)Content{*conv5_block3_3_conv*},
style{*conv2_block3_1_conv,conv3_block3_1_conv,*,
conv4_block3_1_conv, conv5_block3_1_conv}
f)Content{*conv5_block3_3_conv*},
style{*conv2_block2_1_conv,conv3_block2_1_conv,*,
conv4_block2_1_conv,conv5_block2_1_conv}
g)Content{*conv5_block3_3_conv*},
style{*conv2_block3_1_conv,conv3_block3_1_conv,*,
conv4_block3_1_conv,conv5_block3_1_conv}.

This was done with multiple combinations since there wasn't any reference in any paper on which layers to use for a ResNet50. Luckily, the ResNet was much faster compared to the VGG and we could train it easily.

During our experiment, we ended up trying out different weights for the total variation and we found out how crucial it was for our final results [Figure 13].

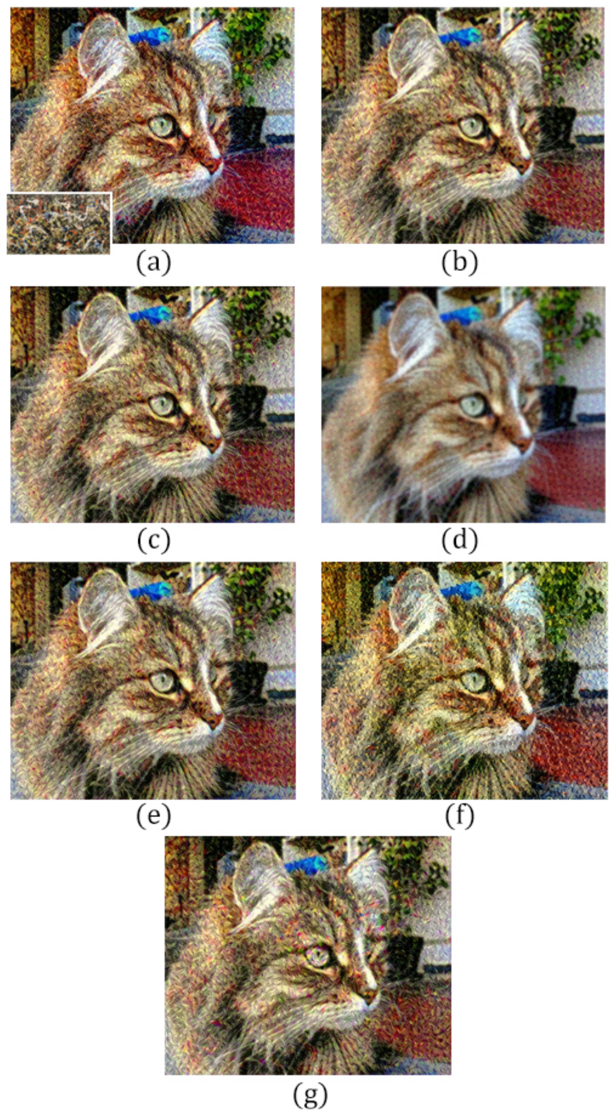

Figure 12. Outputs from ResNet with different layers used for the content. The letters are related to the ones state above, since the caption must be small. The outputs all similar and good looking expect for the (d).

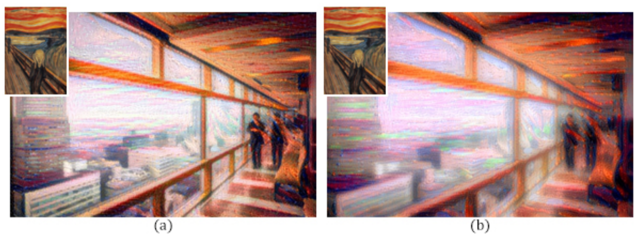

Figure 13. Outputs with different total variation weight comparison. (a) Weight of 10^{-6} (b) Weight of 10^{-5}.

We needed a very low weight as stated before in order to obtain sharp results.

However, even with extensive experiments, our results however were not comparable to the one of the original paper. This is likely because our resources were not comparable to those available to the original authors of the paper.

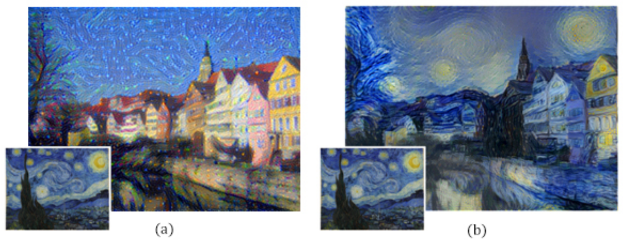

Figure 14. Comparison between the output of our VGG19 and the output of Gatys et.al (a) Our output (b) Gatys et. al. output.

5. Conclusion

In conclusion, we have tested and compared multiple pre-trained network: VGG16, VGG19, ResNet50. The visual performances of the VGG19 were the one we found the most aesthetically pleasing, even if the performance in terms of execution time were the worse of them all. We are not completely satisfied with our results, but we have had difficulties during our project in order to understand better the algorithm along with technical difficulties, since we had to train through Colab which caused multiple limitations by Google since we surpassed the maximum RAM usage.

Overall, we learned how to setup properly a Neural Style Transfer with multiple networks and also we learned about new functionalities from TensorFlow (for example how to update the losses manually and compute gradients through GradientTape).

Some of our suggestions for future extension are:

- Explore the Model-Optimisation-Based for image reconstruction approach rather than the Image-Optimization-Based and compare results - Explore more algorithm to compute the losses and compare results - Explore ways to apply multiple style images to a single content images, for example by applying different styles to different regions of the content image

In the end, we think that Neural Style Transfer is a technique which could refined and researched on even further. It could be applied to apply styles to 3D images or to generate images from just a few aesthetic concepts. Maybe in a future it could be used to create new styles by combining pre-existing styles, which would be great from an artistic point of view. These two possible extensions could be used in many areas, like film-making and game-development.

References

[1] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style, 2015. Neural Style Transfer

on images.

[2] Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, Yizhou Yu, and Mingli Song. Neural style transfer: A review, 2018. Comparison of multiple Neural Style Transfer approaches.

[3] David Mould and Paul L. Rosin. A benchmark image set for evaluating stylization, 2016. NPRGeneral benchmark images to test image stylization algorithms.

[4] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. CNN for large-scale image recognition including VGG.