Exploring Visual Quality in Neural Style Transfer

Jeremy Dale Coronia
College of Computer Studies
De La Salle University
Taft Ave., Malate, Manila
jeremy_coronia@dlsu.edu,ph

***Abstract*—The work of Gatys et al. has demonstrated the possibility of separating the content and and the style of images through recasting the style of one image to the content of another, leading to the field of neural style transfer (NST). Despite issues of efficiency when compared to arbitrary real-time ST, the original work is still considered a gold-standard. In this work, the visual quality of the Gatys et al. work, as well as an implementation of real-time arbitrary ST, will be explored and compared in terms of visual quality. Additionally, an ablation study is performed on the various parameters of the Gatys implementation to investigate their role when generating the stylized outputs. Results show that the Gatys work produced visually appealing images, although in some examples, the arbitrary ST more accurately captures a given art style and at a shorter time. The ablation study also show that the number of layers, content-style ratio, and image dimensions played the largest roles when generating the stylized outputs.**

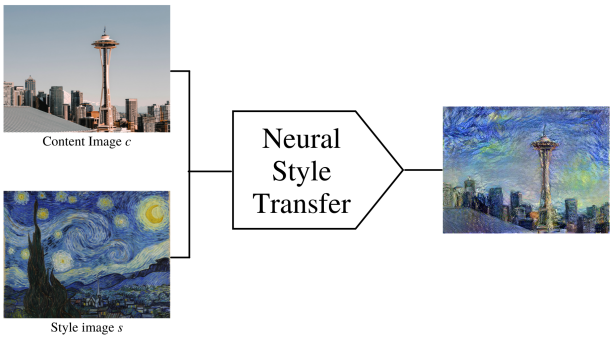

Fig. 1. A representation of transferring the style of a given **style image *s*** to a **content image *c*** to generate a stylized output. The style image is Van Gogh's The Starry Night (1889).

I. Introduction

The seminal work of Gatys et al. [1] showed the capability of convolutional neural networks (CNNs) to separate the content and style information from given images. They demonstrated this by exploiting the learned feature maps of the different CNN layers to recombine the content and the style of two (2) different images into generated stylized outputs. This opened up a new field called neural style transfer (NST). The style images used in their study were well-known classical artworks. In their work, they used the VGG-19 model pre-trained on the ImageNet dataset[1], which has been trained on image classification [2].

Figure 1 below shows a representation of this neural style transfer algorithm. In this example, the style of a [style] image *s*, The Starry Night, is transferred onto a content image *c*.

Despite recent advances in the field, the image-optimization-based algorithm of Gatys et al. [1] is still hailed as a "gold standard" in neural style transfer [3], being visually appealing despite suffering from efficiency issues. Because it is computationally expensive, however, other model-optimization-based methods have been explored, trading speed and flexibility with quality. Previous work on real-time (fast) style transfer include Johnson et al. [4], although the biggest drawback is the failure to perform arbitrary style transfer, requiring one trained network for one style. Alternatively, Dumoulin et al. [5] succeeded in learning new styles without the disadvantage of having to train separate networks for each, although it was found that model size proportionally increases to learned style representations.

The work by Ghiasi et al. [6], meanwhile, which trained a model on 80, 000 paintings, is said to capture and transfer artworks not seen during training. The core idea of this 'arbitrary' style transfer is to essentially be a "one-model-for-all" [3]. This approach is used in this experiment.

Given all this, this work aims to explore the Gatys et al. implementation [1], and compare its output to a real-time arbitrary style transfer algorithm [6]. Visual quality would be qualitatively compared. Moreover, the various parameters of the models would also be explored in an ablation study to investigate which parameters have a significant effect to the output of the image.

This section is thus divided into the following sections. The next section covers the methodology of both approaches discussed above. This is followed by results from experimenting with both style transfer approaches. Finally, the summary, conclusion and directions for future work are reported.

II. Methodology

This section is divided into two (2): the Image-based-Optimization Algorithm [1], and the real-time fast Arbitrary Style Transfer method [6].

A. Image-Based-Optimization Algorithm

A given input image x fed into the VGG-19 (image classification) network is encoded as a set of filter responses of that image as it progresses through the layers of the

[1]https://image-net.org/

CNN. Progression into the deeper layers means progression of more complex filter responses (e.g., whole shapes as opposed to lower-level feature responses like edges). To reconstruct this content representation of the input image x encoded at different hierarchical layers, gradient descent is performed on a white noise image to match another image using the feature responses F^l of x. Essentially, this random white noise image is continually tweaked until it generates the same feature responses as x.

Thus, Equation 1 shows the squared-error loss between two (content) feature representations, where p and x are the original image and generated image, respectively. Meanwhile, P^l and F^l are their respective content representations at layer l.

$$L_{content}(p, x, l) = \frac{1}{2}(\sum_{i,j}(F_{ij}^l - P_{ij}^l)^2 \tag{1}$$

Next, the style representation learned is a multi-scale representation from five (5) different layers, as used by Gatys et al. [1], which computes the correlation among the filter responses. The lower network layers included learn lower-level features such as textures and edges, while the deeper layers, the [style] image structures are more accurately defined, generating smoother stylized outputs.

The feature correlations, as used in the Gatys et al. work, are computed using a Gram matrix $G^l \epsilon R^{N_l x N_l}$. Seen below, Equation 2 shows that the gram matrix G of layer l is the inner product of the feature maps (i, j) of that layer.

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l. \tag{2}$$

Afterward, using another white noise image, the mean-squared distance between the gram matrices are minimized to be able to generate the texture from a given style image s, as shown in Equation 3 where A^l and G^l is the style representations of the original image and the generated image, respectively.

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j}(G_{ij}^l - A_{ij}^l)^2 \tag{3}$$

The final total style loss is shown in Equation 4, where w_l is the loss weight at layer l and E_l is the loss at that layer.

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

Finally, to recast the style of style image s to the content image c, the content representations of the white noise image and the style representations of the artwork are jointly minimized as shown in the loss function in Equation 5. $\overrightarrow{p}$ is the content image and $\overrightarrow{a}$ is the style image.

$$L_{total}(\overrightarrow{p}, \overrightarrow{a}, \overrightarrow{x}) = \alpha L_{content}(\overrightarrow{p}, \overrightarrow{x}) + \beta L_{style}(\overrightarrow{a}, \overrightarrow{x}) \tag{5}$$

This α/β ratio show the weighting factors of the content over style. It describes how much the generated output will match either the content or the style of the artwork. This ratio is explored and discussed further in the next section, specifically, at Figure 7.

Regarding implementation, this work used an open-sourced code from TensorFlow [7], [8] with some modifications made to reflect the experiments below. Different configurations were done with the aid of [1], [9].

B. Real-time Arbitrary Transfer

The work of Ghiasi et al. [6] is motivated by the observation that a style transfer network trained on a wide range of paintings would be able to learn a "rich vocabulary" for effective real-time arbitrary style transfer of any given artwork, even for art styles not seen during training. Their proposed model architecture is shown below in Figure 2.

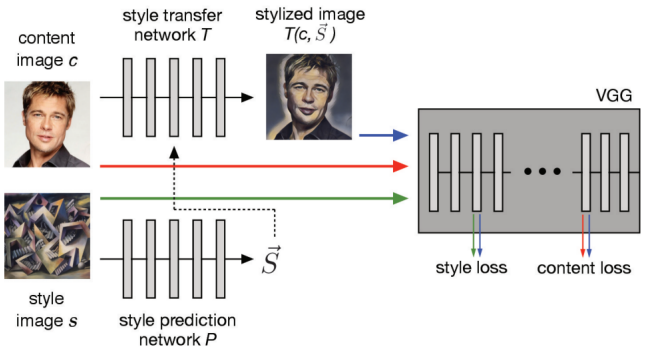

Fig. 2. The proposed arbitrary style transfer model. Image taken from the work of Ghiasi et al. [6].

Their architecture extends the style transfer network by adding a *style prediction network* P that predicts an embedding vector $\overrightarrow{S}$ given a style image s. This gives a set of normalization constants for the style transfer network T, which is built similarly from [5], to generate the stylized output. Essentially, this architecture is able to generalize well even to unknown or unseen painting styles during test time. Both style prediction network P and style transfer network T were trained on the ImageNet set for content images and Kaggle's Painter by Numbers[2] set for style images, which were augmented. It was also trained on the augmented Describable Textures Dataset [10] for style images, which is an image corpus of 5,640 images in 47 categories.

For this experiment, the pre-trained arbitrary image stylization model from TensorFlow Hub was used[3]. The implementation is from an open-source tutorial by TensorFlow Hub [11], [12]. Because the model used is an out-of-the-box model from a Google-backed research project called Magenta, there is limited customizability for these experiments.

[2]https://www.kaggle.com/c/painter-by-numbers
[3]https://tfhub.dev/google/magenta/arbitrary-image-stylization-v1-256/2

III. Results

In this section, the results after experimentation will be discussed.

First, Figure 3 shows the results from the Gatys et al. [1] when mixing a publicly available photo from Unsplash with various artworks. It can be seen that for all outputs, the generated output somewhat retains the art style of the style image. An interesting observation, however, is that for Outputs (A), (B) and (E), there is a large concentration of color on the top-left corner of the image, which may be attributed to the darker coloration and contrast compared to the rest of the topmost section, though this wasn't explored further.

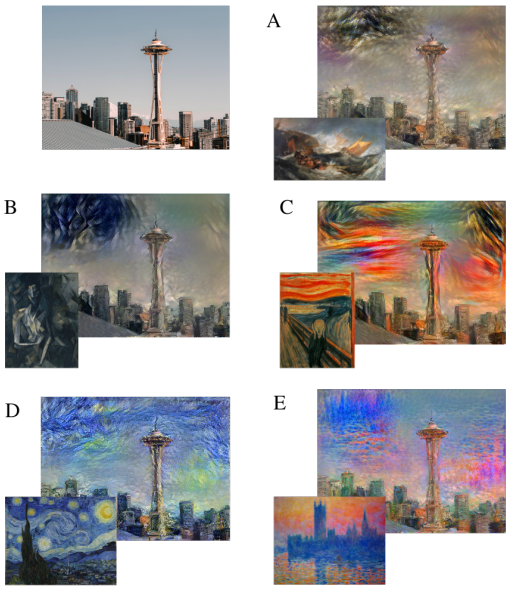

Fig. 3. Sample outputs of style transfer from Gatys et al. [1]. The content image used here is a public image Cody Fitzgerald on Unsplash (2020). The different style images used are as follows: **A** The Shipwreck of the Minotaur by J. M. W. Turner; **B** Seated Nude by Pablo Picasso (1909); **C** The Scream by Edvard Munch (1893); **D** The Starry Night by Vincent Van Gogh (1889); **E** The Houses of Parliament, Sunset by Claude Monet (1903)

Next, Figure 4 shows the difference in generated outputs from both Gatys et al. [1] and Ghiasi et al. [6]. A notable observation here is that the Ghiasi et al. outputs appear more faithful to the style of both artworks. This can be seen more with Example 2: Picasso's Seated Nude (1909) seems to have a more geometric style compared to the more wavy-like styles of Munch's The Scream (1889), and the Arbitrary ST output seems to copy better than the Gatys et al. implementation. The Gatys et al. implementation meanwhile produced mosaic-like stylized outputs. It is also observed that in Example 1, all outputs produced artefacts at the image borders and around the tower.

A notable difference between the two (2) implementations is that the work of Ghiasi et al. produced images at a significantly shorter time than Gatys et al. On a 8-core AMD Ryzen 7 4800H processor, the arbitrary real-time ST managed to produce several outputs in a maximum of just under 1 minute. This is compared to the Gatys et al. work where generating a single stylized output took, at worst, 20 minutes.

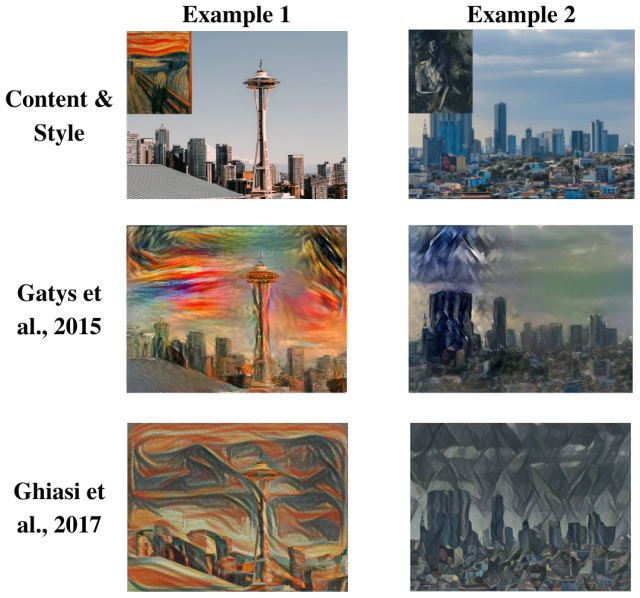

Fig. 4. A comparison of two different style transfer outputs from both Gatys et al. [1] and Ghiasi et al. [6].

To further look at the difference between either implementation, the image dimensions were also modified, namely: 256, 384, 516, 1024. Figure 5 shows the outputs in those different settings. It should be noted here that both content and style image dimensions were rescaled. The work of Ghiasi et al. [6] used the image dimension (256, 256).

For the Gatys et al. implementation, when the images were rescaled to a lower dimension, it more faithfully recreated the art style, owing to the feature maps capturing more of the art style, as compared to a larger dimension where the algorithm could not sufficiently learn the art style. This is also true for the Ghiasi et al. implementation, though another observation is that the outputs looks almost patchwork-based, most prominently when the image dimensions were rescaled to 516 or 1024. The default dimension used in these experiments was 516. All in all, the results highlight the need to properly configure image dimensions before training.

To further explore the capabilities of the Gatys et al. implementation, various parameters were tweaked to see their effects on the output.

Firstly, the total variation loss was explored. Total variation loss is a regularization term to help reduce high frequency artifacts in the stylized image. This can be seen in Figure 6, where outputs with lower total variation loss show more "roughness" or image artifacts, while higher values show a smoother output. For this experiment, 30 is the default value used.

Next, the number of [style] layers and the content-style ratio of the different were experimented with. Figure 7 shows the

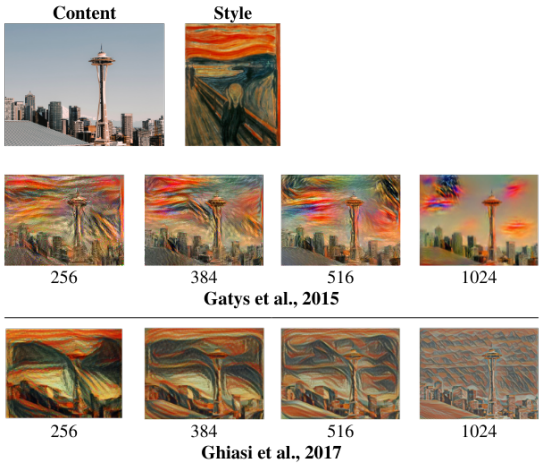

Fig. 5. A comparison of outputs from both Gatys et al. [1] and Ghiasi et al. [6] when changing the dimensions of the content and style images.

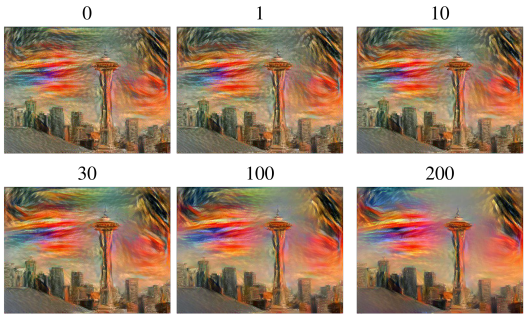

Fig. 6. The results of style transfer when the total variation loss is changed, which affects the smoothness of the stylized output. 30 is the baseline used in this report.

results. The rows show the number of layers while the columns indicate the relative weightings (α/β). For this experiment, the content weight was kept constant at $1x10^4$, and the default style weight used in the other experiments was $1x10^{-2}$.

The results show that a larger content-style ratio preserves more of the content image at the cost of the style. The inverse is also true: the content image becomes more distorted as the ratio value gets lower, but the style is more reflected.

Alternately, in observing the role of the number of layers relative to the content-style ratio, it can be seen that between four (4) to five (5) layers, there is little noticeable difference beyond more smoothing and removal of artifacts, such as around the image borders. Similarly, the content-style ratio show not much differences in generated outputs starting from $1x10^4$. It would be interesting to see the results to a significant change in the style ratio.

Finally, regarding the number of [style] layers used, it can be observed that when using only one (1) layer, the VGG-19 network cannot adequately learn the style representations of The Scream. A bigger style weight somewhat alleviates that, showing some understanding of the color and textures of the image, although still not enough nor is it visually appealing. Starting from the three (3) layers and more, the model defines the style representations with more fidelity. It should be noted that the Gatys et al. study [1] used 5 layers to define the style representation, which is the same default number of layers used here in other experiments.

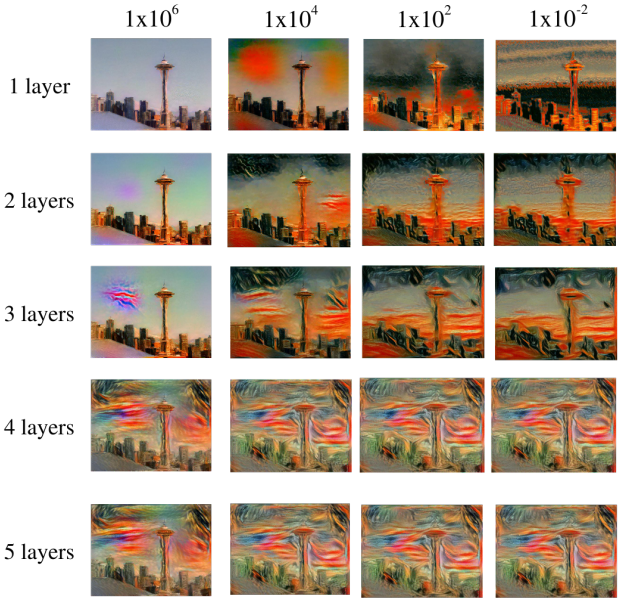

Fig. 7. The results of style transfer when changing the layers or the relative weightings between style and weight loss. The rows indicate the number of layers used while the columns indicate the content-style ratio.

IV. Conclusion

In this report, the original implementation of neural style transfer [1], as well as a real-time arbitrary style transfer [6] was performed and documented. An ablation study was also performed where the various parameters in the Gatys et al. implementation were explored to determine the degree of impact on the resulting stylized output. Based from qualitative observations, it was found that the number of layers, the relative weightings, and the dimensions played a large part in the stylized output. Although, the results also show that there is little noticeable differences between using four (4) layers or the complete five (5) layers as used originally.

A comparison was also made between the Gatys et al. implementation [1], as well as the arbitrary style transfer by Ghiasi et al. [6]. Experiments show decreased quality of output for the arbitrary style transfer, especially with complex patterns; however, a main advantage from arbitrary style transfer is its capability to more efficiently perform style transfer as compared to the other implementation. Still, the stylized output from [1] is still qualitatively better than [6].

For future work, it would be interesting to perform more experiments regarding arbitrary style transfer, as well as explore more evaluation measures for style transfer, particularly, quantitative evaluation (e.g., efficiency in terms of training

time or time to generate an output was not explored here). Experimenting with different artistic styles may also be an interesting direction to take, such as with pop art, low-poly styles or more abstract arts. Including other style transfer approaches from other studies may also be undertaken, as well as investigating photorealistic style transfer, as both approaches discussed here are non-photorealistic.

REFERENCES

[1] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," *arXiv preprint arXiv:1508.06576*, 2015.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[3] Y. Jing, Y. Yang, Z. Feng, J. Ye, Y. Yu, and M. Song, "Neural style transfer: A review," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 11, pp. 3365–3385, 2019.

[4] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," 2016.

[5] V. Dumoulin, J. Shlens, and M. Kudlur, "A learned representation for artistic style," 2017.

[6] G. Ghiasi, H. Lee, M. Kudlur, V. Dumoulin, and J. Shlens, "Exploring the structure of a real-time, arbitrary neural artistic stylization network," *arXiv preprint arXiv:1705.06830*, 2017.

[7] TensorFlow, "Neural style transfer," 2018. [Online]. Available: https://github.com/tensorflow/docs/blob/master/site/en/tutorials/ generative/style_transfer.ipynb

[8] F. Nolan, "Neural algorithm of artistic style: A modern form of creation," 2019. [Online]. Available: https://towardsdatascience.com/a-neural-algorithm-of-artistic-style-a-modern-form-of-creation-d39a6ac7e715

[9] S. Singla, "Experiments on different loss configurations for style transfer," 2017. [Online]. Available: https://towardsdatascience.com/experiments-on-different-loss-configurations-for-style-transfer-7e3147eda55e

[10] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, "Describing textures in the wild," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3606–3613.

[11] TensorFlow Hub, "Fast style transfer for arbitrary styles," 2019. [Online]. Available: https://github.com/tensorflow/hub/blob/master/examples/ colab/tf2_arbitrary_image_stylization.ipynb

[12] O. G. Yalçin, "Fast neural style transfer in 5 minutes with TensorFlow Hub & Magenta," 2020. [Online]. Available: https://towardsdatascience.com/fast-neural-style-transfer-in-5-minutes-with-tensorflow-hub-magenta-110b60431dcc