Assignment 6 & 7 : Neural Style Transfer

Team Name : The Salvator Brothers
School of Engineering and Applied Science, Ahmedabad University

Computer Vision - CSP502, Winter 2021

1st Kirtan Kalaria
AUL202005
Ahmedabad University
Ahmedabad, India
kirtan.k@ahduni.edu.in

2nd Manav Vagrecha
AU1841022
Ahmedabad University
Ahmedabad, India
manavkumar.v@ahduni.edu.in

CONTENTS

***Abstract*—This work presents a neural algorithm of artistic style, known as Neural Style Transfer, that can separate and recombine the image content and style of natural images. The algorithm allows us to produce new images of high perceptual quality that combine the content of an arbitrary photograph with the appearance of numerous well-known artworks. This work provides insights into the deep image representations learned by different Convolutional Neural Network architectures viz. VGG-16, VGG-19, and ResNet50.**

***Index Terms*—Neural Style Transfer, Images with Artistic Style, Convolutional Neural Networks, Content-Style Images.**

I. INTRODUCTION

Neural style transfer is an optimization technique used to take two images—a content image and a style reference image (such as an artwork by a famous painter)—and blend them together so the output image looks like the content image, but "painted" in the style of the style reference image. This is implemented by optimizing the output image to match the content statistics of the content image and the style statistics of the style reference image. These statistics are extracted from the images using a convolutional network. The principle of neural style transfer is to define two distance functions, one that describes how different the content of two images are, $\mathcal{L}_{content}$, and one that describes the difference between the two images in terms of their style, $\mathcal{L}_{style}$. Then, given three images, a desired style image, a desired content image, and the input image (initialized with the content image), we try to transform the input image to minimize the content distance with the content image and its style distance with the style image.

II. LITERATURE REVIEW

Let us suppose $\overrightarrow{p}$, $\overrightarrow{a}$ and $\overrightarrow{x}$ be the content image, style image and the output image generated after style transfer. Also assume $\mathcal{P}_l$ and $\mathcal{F}_l$ be the feature vectors of the content image and the resultant image for the l^{th}. F^l_{ij} be the activation of i^th filter at position j of the layer l.

Now we have,

$$\mathcal{L}_{content}(\overrightarrow{p}, \overrightarrow{x}, l) = \frac{1}{2}\sum_{i,j}(\mathcal{F}^l_{ij} - \mathcal{P}^l_{ij})^2$$

To obtain a representation of the style of an input image, we use a feature space designed to capture texture information. This feature space can be built on top of the filter responses in any layer of the network. It consists of the cor-relations between the different filter responses, where the expectation is taken over the spatial extent of the feature maps. These feature correlations are given by the Gram matrix $G^l \epsilon R^{N_l \times N_l}$, where G^l_{ij} is the inner product between the vectorized feature maps i and j in layer l

$$\mathcal{G}^l_{ij} = \sum_k \mathcal{F}^l_{ik}\mathcal{F}^l_{kj}$$

Also, the contribution of each layer to the loss can be determined by

$$E^l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_i^l j - A_i^l j)^2$$

and finally, the loss of the style images will be

$$\mathcal{L}_{style}(\overrightarrow{a}, \overrightarrow{x}) = \sum_{l=0}^{L} w_l E_l$$

Usually, there does not exist a perfectly synthesized image that matches both constraints of combining the content of one image with the style of another. However, assuming a linear combination between the loss functions for content and style respectively, we can smoothly regulate the emphasis on either reconstructing the content or the style. Thus,

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

where α and β are the weighting factors for content and style reconstruction, respectively. The gradient with respect to the pixel values $\frac{\partial \mathcal{L}_{total}}{\partial x}$ can be used as input for some numerical optimisation strategy as

$$\overrightarrow{x} := \overrightarrow{x} - \lambda \frac{\partial \mathcal{L}_{total}}{\partial \overrightarrow{x}}$$

III. Algorithm and Process-Flow

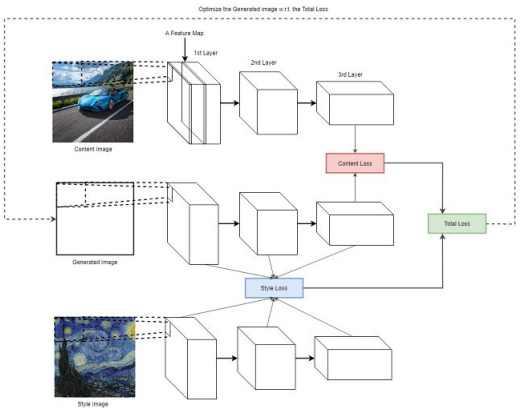

Fig. 1. Process flow

1) First content and style features are extracted and stored.
2) The style image $\overrightarrow{a}$ is passed through the network and its style representation A_l on all layers included are computed and stored.
3) The content image $\overrightarrow{p}$ is passed through the network and the content representation P_l in one layer is stored.
4) Then a random white noise image $\overrightarrow{x}$ is passed through the network and its style features G_l and content features F_l are computed.
5) On each layer included in the style representation, the element-wise mean squared difference between G_l and A_l is computed to give the style loss $\mathcal{L}_{style}$. Also the mean squared difference between F_l and P_l is computed to give the content loss $\mathcal{L}_{content}$.
6) The total loss $\mathcal{L}_{total}$ is then a linear combination between the content and the style loss.
7) The derivative of total loss with respect to the pixel values can be computed using error back-propagation. This gradient is used to iteratively update the image $\overrightarrow{x}$ until it simultaneously matches the style features of the style image $\overrightarrow{a}$ and the content features of the content image $\overrightarrow{p}$.

IV. Architectural Overview

Architectures for VGG-16 and VGG-19 :

- The first two layers are convolutional layers with 3×3 filters, and in the first two layers we use 64 filters so we end up with a 224×224×64 volume because we're using same convolutions (height and width are the same).
- So, this (CONV64)×2 represents that we have 2conv layers with 64 filters. The filters are always 3×3 with stride of 1 and they're always implemented with the same convolutions.
- Then, we use a max pooling layer which will reduce height and width of a volume: it goes from 224×224×64 down to 112×112×64.
- Then we have a couple more conv layers. Here we use 128 filters and because we use the same convolutions, a new dimension will be 112×112×128.
- Then, a pooling layer is added so new dimension will be 56×56×128.
- 2conv layers with 256 filters
- The pooling layer
- A few more conv layers with 512 filters (3 in VGG-16 where as 4 in case of VGG-19)
- A pooling layer
- A few more conv layers with 512 filters
- A pooling layer
- At the end we have final 7×7×512 into Fully Connected layer (FC) with 4096 units, and in a softmax output one of a 1000 classes

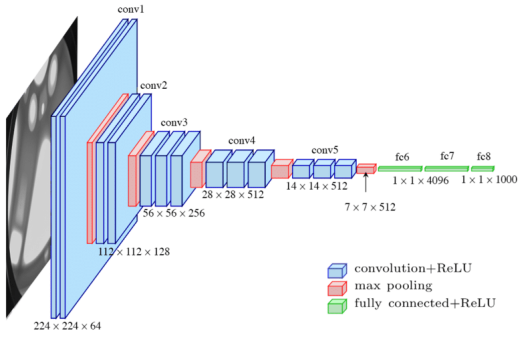

Fig. 2. Architecture: VGG-16

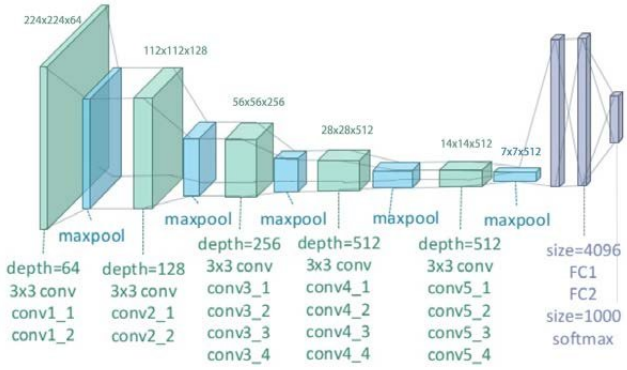

Fig. 3. Architecture: VGG-19

V. Implementation

First, we set a function to load images from the path and preprocess it by resizing it and setting the dimension according to our requirements. We also define a function to plot the image in a proper format.

Then we set up the image preprocessing function to preprocess it according to the chosen CNN architecture model's specifications. We also set up a deprocessing function to reverse the preprocessing.

Next, we specify the representational layers for style and content, for each of the three models and define a function to get the style and content features for a pair of content and style image pair based on the defined representational layers.

Following that, we build the chosen model and configure it for use. Then, we define the content, style, and total loss functions and compute the losses and gradients.

After that, we set up the optimization function to return the best image based on the minimum total loss.

Finally, we define the final function *show_results(...)* that does the runs process and outputs results. We just need to call this function with the content image and style image paths and an integer (0, 1 or 2) to specify the model to be used. Further, optionally we can specify the hyperparameters content weight, style weight and learning rate to tune the results.

We have carried out the style transfer process using 4 content images (New York Skyscrapers, Grand Canyon, Hritik Roshan, and Despicable Me), 5 style images (Van Gogh's Starry Night, Edvard Munch's The Scream, Linguist: A Trippy Image, Honeycomb, and Mandala) and 3 pre-trained models (VGG16, VGG19, and ResNet50). All this is done for content images of the resolution 1024×1024×3. The style image resolution was unchanged and higher than 1024×1024. Additionally, we have done the process for the New York Skyscraper content image of 128×128×3 resolution as well for all styles and with all models to check the effect of resolution.

The idea is to get a variety for the style and content. For content, we have the categories: urban, natural, human and animated.

VI. Input Images

A. Content Images

Fig. 4. New York Skyscrappers

Fig. 5. Grand Canyon Arizona

Fig. 6. Hritik Roshan

Fig. 7. Despicable Me

B. Style Images

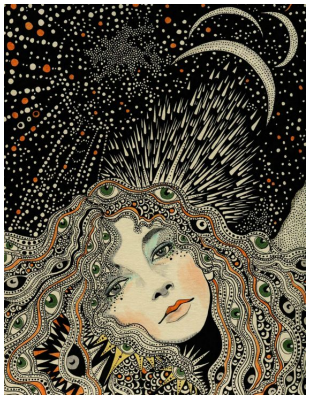

Fig. 8. Trippy Linguist

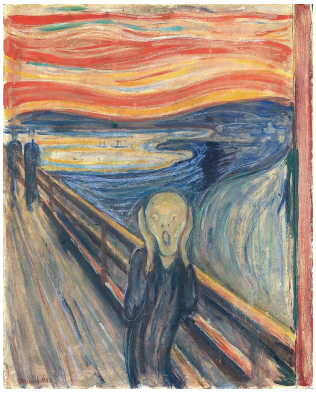

Fig. 9. The Scream

Fig. 10. The Starry Night

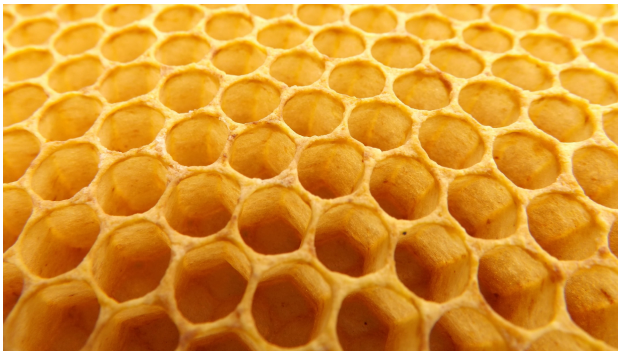

Fig. 11. Honeycomb

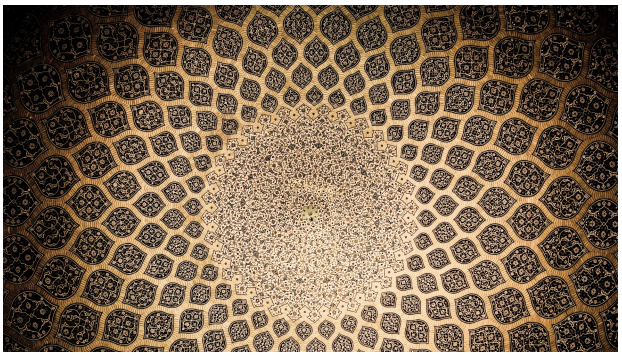

Fig. 12. Mandala

VII. Results

In totality, we have (4×5×3) + (1×5×3) = 75 result images but we will display only 16 of them here to cover the jist of the results. The other results can be seen in the Colab Notebook with saved outputs here.

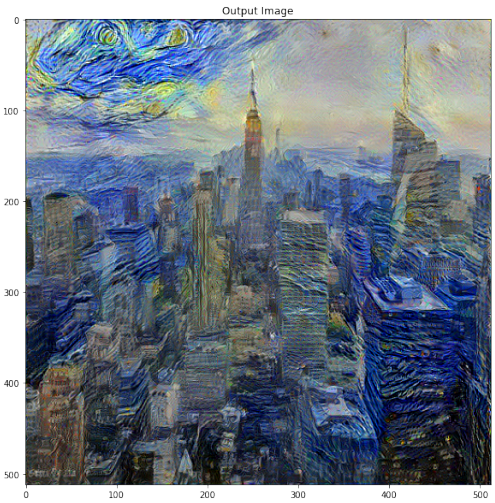

Fig. 13. New York × Starry Night; VGG-16

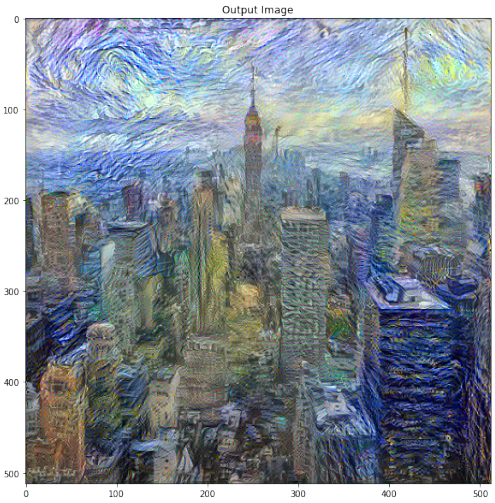

Fig. 14. New York × Starry Night; VGG-19

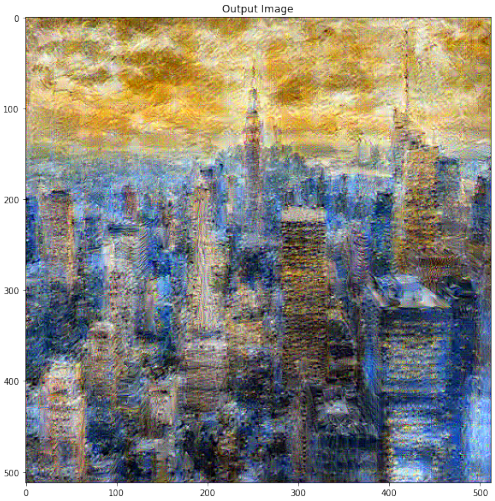

Fig. 15. New York × Starry Night; ResNet50

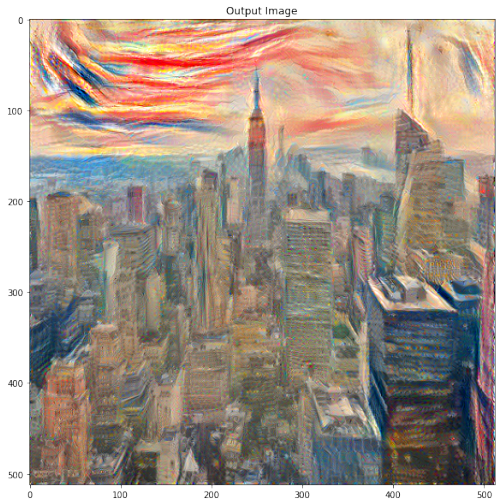

Fig. 16. New York × Scream; VGG-16

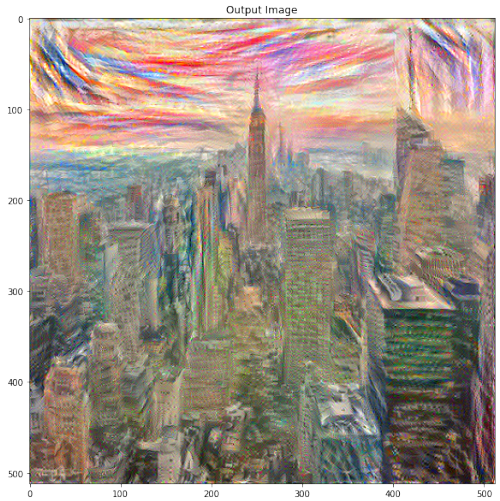

Fig. 17. New York × Scream; VGG-19

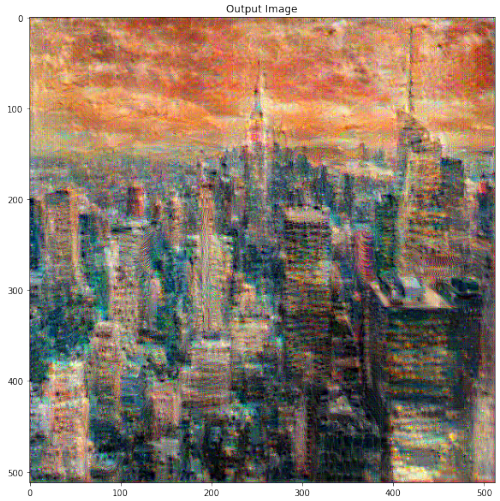

Fig. 18. New York × Scream; ResNet50

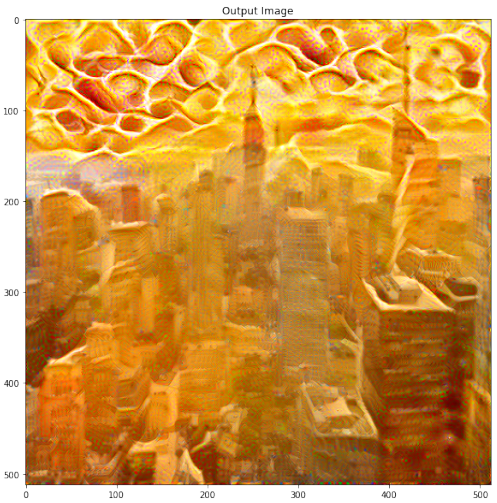

Fig. 19. New York × Honeycomb; VGG-16

Fig. 20. New York × Honeycomb; VGG-19

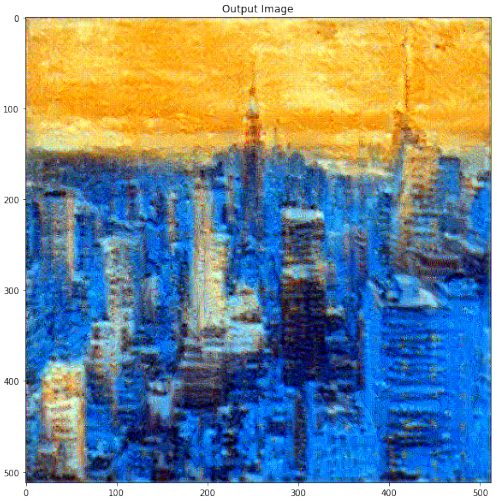

Fig. 21. New York × Honeycomb; ResNet50

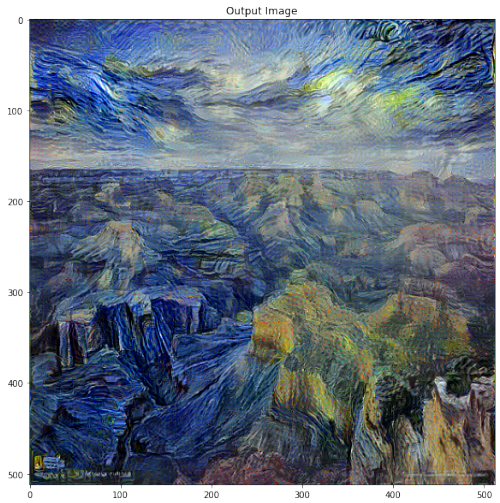

Fig. 22. GrandCanyon × StarryNight; VGG-16

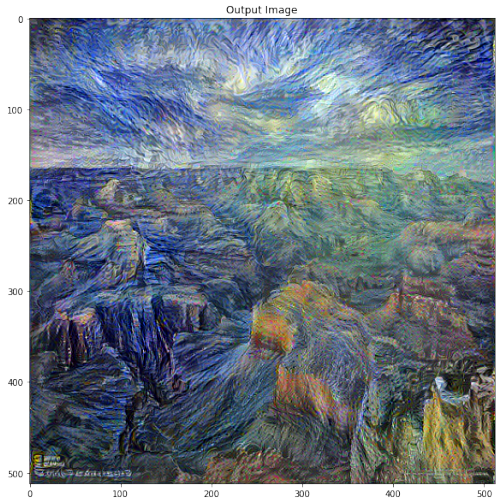

Fig. 23. GrandCanyon × StarryNight; VGG-19

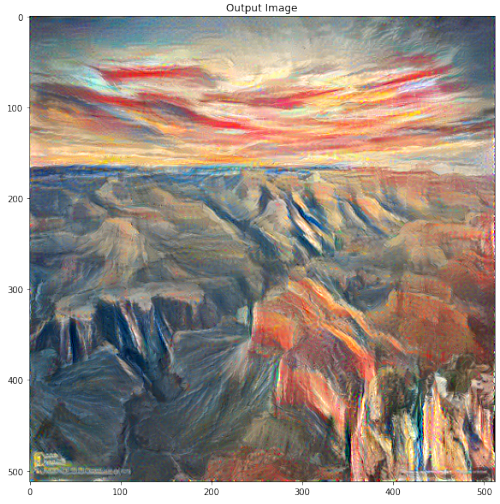

Fig. 24. GrandCanyon × Scream; VGG-16

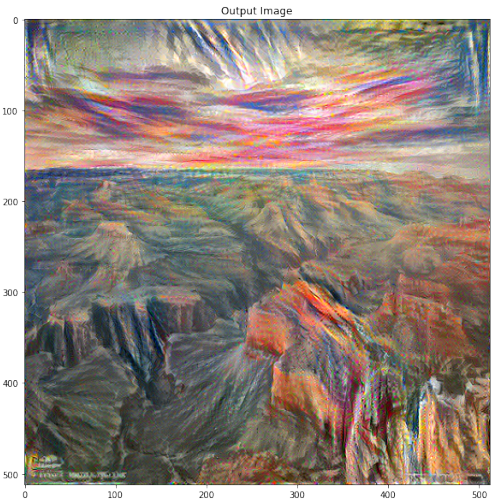

Fig. 25. GrandCanyon × Scream; VGG-19

Fig. 26. DespicableMe × Starry Night; VGG-16

Fig. 27. Hritik Roshan × Starry Night; VGG-16

Fig. 28. NewYork (128 Low Res) × Scream; VGG-16

Fig. 29. NewYork (128 Low Res) × Scream; VGG-19

Fig. 30. NewYork (128 Low Res) × Scream; ResNet50

VIII. Discussion

Neural style transfer is more art than science. The quality of result in style transfer, is largely a subjective matter. The expectations can vary from person to person, so does the likeability of the result. Much also depends on the difference between the expectation and the result and the flexibility of the viewer. So, we will discuss the following according to our point of view.

A. Effect of model architecture

VGG16 and VGG19 have much better performance than ResNet50. VGG models succeed in capturing the style much better than ResNet50, even though they were trained on the same ImageNet dataset. Although VGG16 and VGG19 perform similarly, they have some noticeable differences.

VGG19 produces relatively smoother images but slightly deforms the structure of objects. On the other hand, VGG16 captures exact style elements (type of higher complexity features) from the style image better, for example in Fig. 22, we can see the yellow spots in the blue strokes exactly like in the original style image, while VGG19 captures lower level features like the strokes of alternating shades of blue better. This also contributes to the smoothness of the result image as lesser complex features are blended better in VGG19, but at the slight cost of exact stylistic elements. We can't declare one better than the other; the more suitable model depends on the what type of stylistic elements (i.e. features of the style image) we want to capture, rendering it a subjective choice.

Meanwhile, ResNet50, irrespective of the choice of style and content layers (which is an extremely cumbersome process), captures content much better but fails to capture complex style features satisfactorily. However, in its own way, the resulting image is not unpleasant and looks more like an oil painting. On the other hand, VGG images look like crayon or water color paintings.

It would be interesting to see the effect of using representation layers from VGG16 or VGG19 as well as ResNet50. From literature review, we even found some people who have tried these models pre-trained on datasets other than ImageNet. They observed that the results we get are not due to the dataset used but due to the architecture of the models. In other datasets, they found the same differences in performance and results. It appears that because ResNet50 is very complex, the style features are distributed in a very large number of layers, reducing the variation covered in each block. Think of it as 'feature density'. Hence, ResNet50, and by extension, all ResNet models fail to capture complex styles without significant loss in content. The difference in performance of the two VGG models can also be explained by the same logic. However, very low architecture complexity may result in the style and content not being captured well. Hence there is a sweet spot for model complexity and VGG16 and VGG19 lie close to it.

Also, as the model complexity increases, we have to include more layers starting from the beginning to cover larger scale style features, which results in other set of issues.

B. Effect of content and style loss weights

We isolate style and content in the style transfer algorithm but it is not possible to do it perfectly. At best, we create two sets of representations - one with more content representation than style representation and the other the other way around. We rely on the difference in each representation to effective represent the style and the content.

Hence, we cannot perfectly combine the style from the style image and content from the content image. However, we can make different choices of the representational layers to suit our need. And after that we calculate the style loss and content loss.

But in order to optimise the result, we need just one loss that includes the style and content losses. We can tune the process to balance the emphasis style and content by controlling the weights of style and content losses in the total loss.

Higher content loss weight will help preserve the visual structure of the content image but will compromise the style and vice versa for higher content loss weight. There is a clear trade-off. The key is to find the right balance by adjusting the ratio of the two.

C. Effect of content and style image type

Natural images tend to be smoother than urban or artificial images, including animated images. Moreover, the former has a lesser geometric perfection than the latter, like sharp edges and corners. Thus the loss of content is more prominent in the latter.

Moreover, certain pairings of style and content images tend to perform better. There is no concrete logic to explain this. It just is, just like wine and cheese pairings. It is also observed that natural content images perform better with natural style images like artwork.

D. Effect of content image resolution

It appears that low resolution content images result in smoother result images. It has lesser noise like fewer mono-color blobs and patches.However, content structures, especially finer structures are also compromised marginally in case of low resolution images.

The better choice depends on the type of style features we are looking for in the style image. If we want more localized features, high resolution content images are better.

We believe the same could be achieved with regularisation layers in the models, especially blurring or smoothing layers.

IX. Conclusion

In this work, we have implemented neural algorithm for Neural Style Transfer using few pre-trained classical CNN architectures such as VGG16, VGG19 and ResNet50 and their results have been discussed and compared. Conclusively, this process is not suited for batch processing. Rather, it is to be implemented as a customized process by choosing representation layers and hyperparameters for content and style image pairs to get the desired result. We can leverage the inferences and learnings from the discussion of results and use it to improve the process in many ways. Since it is a relatively newer topic, very less work has been done in this area but there is a large scope of improvement, which again, lies in the interdisciplinary intersection of art and computer vision, requiring sufficient proficiency in both fields.

References

[1] L. A. Gatys, A. S. Ecker, and M. Bethge, “A Neural Algorithm of Artistic Style,” arXiv.org, 02-Sep-2015. [Online]. Available: https://arxiv.org/abs/1508.06576. [Accessed: 28-Mar-2021].

[2] S. Desai, “Neural Artistic Style Transfer: A Comprehensive Look,” Medium, 14-Sep-2017. [Online]. Available: https://medium.com/artists-and-machine-intelligence/neural-artistic-style-transfer-a-comprehensive-look-f54d8649c199. [Accessed: 28-Mar-2021].

[3] TensorFlow, “Neural Style Transfer: Creating Art with Deep Learning using tf.keras and eager execution,” Medium, 27-Sep-2018. [Online]. Available: https://medium.com/tensorflow/neural-style-transfer-creating-art-with-deep-learning-using-tf-keras-and-eager-executio [Accessed: 28-Mar-2021].

[4] T. Ganegedara, “Intuitive Guide to Neural Style Transfer,” Medium, 05-Dec-2020. [Online]. Available: https://towardsdatascience.com/light-on-math-machine-learning-intutitive-guide-to-neural-style-transfer-ef88e46697ee [Accessed: 28-Mar-2021].

[5] N. Kasten, “Art & AI: The Logic Behind Deep Learning ’Style Transfer’,” Medium, 12-Mar-2020. [Online]. Available: https://medium.com/codait/art-ai-the-logic-behind-deep-learning-style-transfer-1f59f51441d1. [Accessed: 28-Mar-2021].

[6] Leon A. Gatys, ”Image Style Transfer Using Convolutional Neural Networks”, . [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Gatys_Image_Style_Transfer_CVPR_2016_paper.pdf. [Accessed: 28-Mar-2021]