Neural Style Transfer Using Convolutional Neural Networks

Clifton Ren Ming Mak
29439701
Electrical & Computer Systems Engineering, School of Engineering, Monash University, Malaysia
cmak0003@student.monash.edu

1. Introduction

Paintings are a universal form of art that can convey the artist's message and is widely appreciated by many people. Many famous paintings have been created, that includes the 'Mona Lisa', Kandinsky's Compositions, 'The Starry Night' and many more. There have been plenty of research and studies done to try and effectively turn images into synthetic artworks [1]. Neural style transfer (NST) on images with convolutional neural networks (CNN) was famously introduced by Gatys, et al. [2] in 2016 and revolutionized the idea of reproducing painting styles on natural images. As discussed in the paper, a limitation in the method proposed by Gatys et al. is that only low-level image features of the target image was used for the texture transfer. In [3], the authors proposed a multi-convolutional-learning technique for photographic painting style transfer on non-photorealistic and photorealistic rendering type images. And in [4], the authors proposed a method of maintaining the target image structure during style transfer by segmenting the objects and extracting the soft semantic masks from style and target image. As the research in this field using CNN is fairly new, there are not many papers clearly discussing on the effects of changing certain hyperparameters in the algorithm. Therefore, in this research, the aim is to implement the algorithm introduced by Gatys et al. and explore how adding individual layer weights to the feature extraction will affect the final results.

The rest of the paper is organized as follows. Section 2 discusses the NST algorithm, and the changes made. Section 3 presents the results of the changes made and provides some analysis. And lastly, Section 6 concludes the paper.

2. Method

From reviewing the algorithm proposed by the Gatys et al. in [2], the framework of the NST using CNN was recreated and is shown in Figure 1. The framework involves 2 main stages – extracting the image features and finding the losses from the generated image.

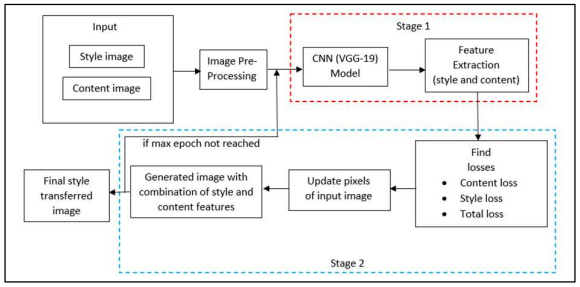

Figure 1: Overview of CNN neural style transfer framework

In the input stage, the style image used is Kandinsky's Composition 7 and the content image is an image of a cat. Both images are then preprocessed by resizing the style and content images such that both have the same size of 512x512.

2.1 Stage 1

In this stage, the style and content will have its features extracted by the CNN model and subsequently obtain the style and feature representation. The CNN model used in the experiment is the VGG-19 model, which is a pretrained network and is 19 layers deep – 5 convolutional blocks. The fully connected layers are not needed as no classification process is needed. As suggested by the authors in [2], the average pooling layer was used in place of the max pooling layers in the CNN model to give better results.

For the style image, the features were extracted from relu1_1, relu2_1, relu3_1, relu4_1 and relu5_1. The features extracted in the higher layers

will lose more pixel information but preserve more high-level content. Features are extracted from all 5 layers to transfer more details of the style to the content image. As for the content image, the features were extracted from the relu3_2 layer as opposed to the suggested conv4_2 layer. Only 1 layer is extracted as the desired outcome is to have more style features and details to be transferred over to the content image. It was suggested in [5] to use relu3_2, as at the lower layers, the network still focusses on the overall feature arrangement rather than individual pixels. This means that the generated image will lose more of its content. Another thing to note is that, as opposed to most papers, the features are extracted after the relu layers rather than the convolutional layers. This was suggested by [5] but it did not provide much improvement.

After the layers were extracted, individual weights were added to the extracted features from the style layers. For each testing, a higher weightage is placed at different relu layers. A test without any individual weights was also performed to act as a benchmark case.

2.2 Stage 2

In stage 2, initially, the generated image will be an empty array. Using the generated image, the content, style, and total loss can be obtained. The content loss is the loss between the content of the generated image and the content of the original content image from the relu3_2 layer. The style loss on the other hand, requires first, the Gram matrix to be computed. According to [5], the Gram matrix is a measure of the correlations between different feature maps in the same layer and is a reasonable measure for texture. The style loss is then found by finding the difference between the Gram matrices of the generated image and the original style image. The total loss is then the sum of the style and content losses but multiplied by some weights. A higher weightage is used on style loss than content loss to emphasize more on the style features. By using an optimizer, the pixel data of the generated image will be updated based on the back propagation of the total loss in each epoch. The process than repeats until the max epoch is reached. The hyperparameters used are summarized in the Table 1. Note that L1 loss was used as opposed to the suggested L2 loss (MSE), as the L1 tolerates more outliers which can improve visual results, as explained in [5].

Table 1: Summary of training hyperparameters

Hyperparameter	Value
Learning rate	0.06
Optimizer	Adam
Loss function	L1
Content loss weight	0.4
Style loss weight	1e6

3. Results and discussion

The results of changing the individual weights of the style layers are discussed in this section. Table 2 summarizes the test cases performed in this experiment. In each test case, a layer is selected to have a weightage of 0.8, while the rest of the layers will have a weightage of 0.2.

Table 2: Summary of test cases

Test Case	Layer with higher weightage
Benchmark	All same weightage of 0.2
1	Relu1_1
2	Relu1_2
3	Relu1_3
4	Relu1_4
5	Relu1_5

Figure 2 shows the generated images after 100, 300 and 1900 epochs for the different test cases. As observed in the images, in all the test cases, focusing on the cat, the content image (cat) is more defined as compared to without adding individual weights. This is because all the weights added are less than 1. Therefore, it is reducing the style content in the generated images.

As the higher weights are placed in the higher style layers, the style background in the generated image becomes more pixelated, as the loss in that layer is emphasized more during optimization. This aligns with the fact that at higher layers, the overall feature arrangements are prioritized over the individual pixels. In test case 4 and 5, the style backgrounds are the least defined as it has become very pixelated and can no longer be clearly visible in the generated image. Comparing test case 4 and

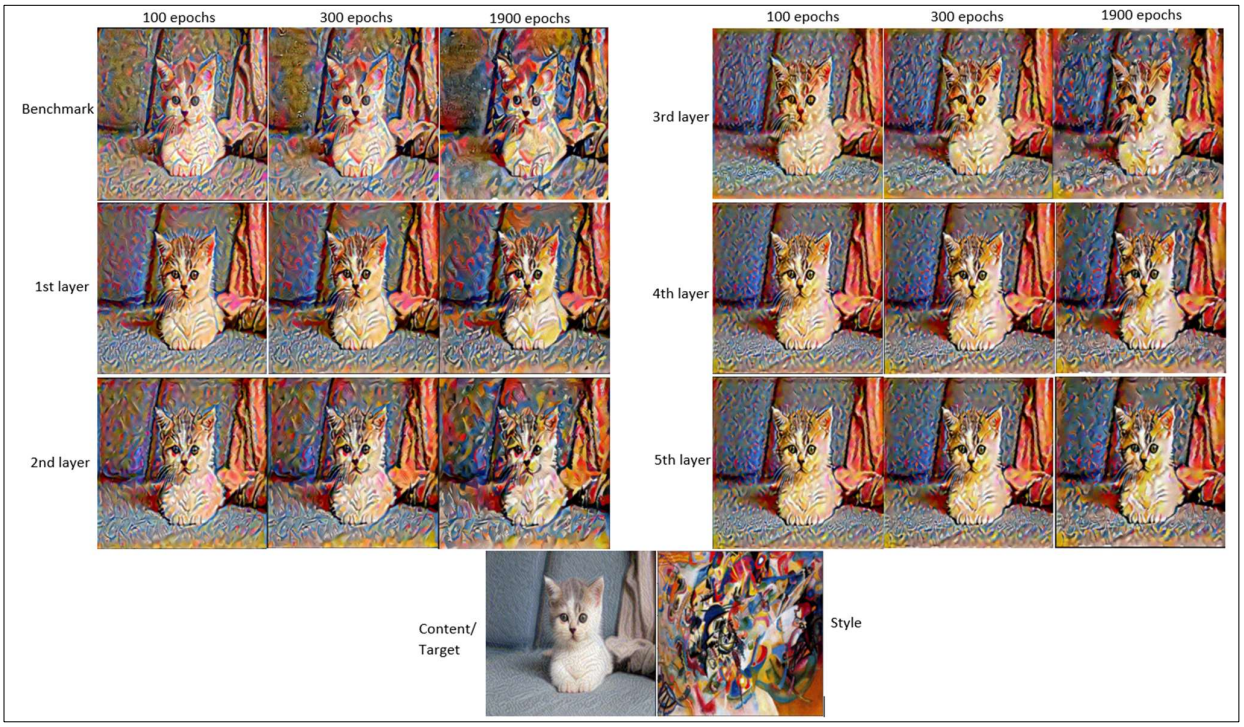

Figure 2: Generated images at 100, 300 and 1900 epochs for different test cases - adding 0.8 individual weight to one layer and 0.2 to the rest of the layers (at diferent selected layers in each test case)

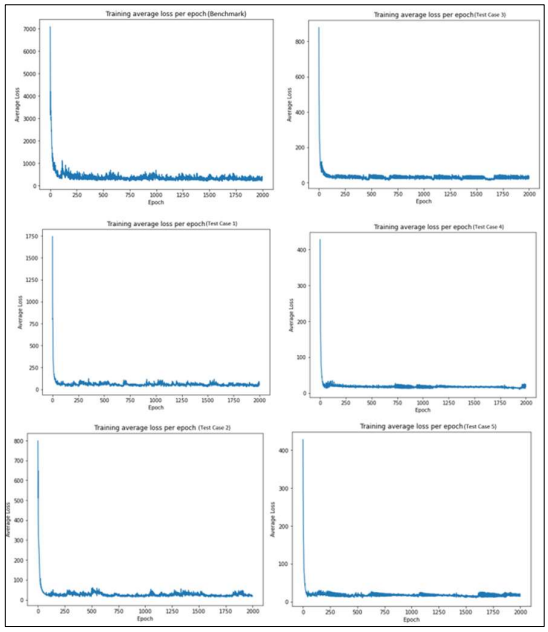

Figure 3: Plot of average total loss for each test cases

5 however, there isn't much of a difference in the final generated image. Based on visual observation, test case 2 yields the best results in terms of balance between the style and content image.

Figure 3 shows the plot of average total losses for all the test cases and benchmark. It can be observed that the difference in the final and initial loss in the benchmark is approximately 6500. In test case 1, it is approximate 1625. In test case 2 and 3, it is approximately 750. And lastly in test case 4 and 5, it is approximately 350. A bigger total loss in this case would indicate a bigger loss in content features and higher gain in style features. This is reflective of the results obtained as the benchmark case with the greatest total loss, is the case with the least defined content features.

4.0 Conclusion

NST with CNN is a fairly new and ongoing research. Different hyperparameters can be tuned to achieve a desired outcome. In this paper, an experiment to include individual weights at the style layers was performed. The results have shown that using weights less than 1 can help preserve the content features better. Also, putting the highest weight (0.8) in the 2nd layer gives the best balance between style and content features.

References

[1] Y. Jing, Y. Yang, Z. Feng, J. Ye, Y. Yu, and M. Song, "Neural Style Transfer: A Review," *IEEE Trans Vis Comput Graph,* vol. 26, no. 11, pp. 3365-3385, 2020, doi: 10.1109/TVCG.2019.2921336.

[2] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image Style Transfer Using Convolutional Neural Networks," vol. 2016-, ed: IEEE, 2016, pp. 2414-2423.

[3] A. Khan, M. Ahmad, N. Naqvi, F. Yousafzai, and J. Xiao, "Photographic painting style transfer using convolutional neural networks," *Multimedia tools and applications,* vol. 78, no. 14, pp. 19565-19586, 2019, doi: 10.1007/s11042-019-7270-8.

[4] H.-H. Zhao, P. L. Rosin, Y.-K. Lai, and Y.-N. Wang, "Automatic semantic style transfer using deep convolutional neural networks and soft masks," *The Visual computer,* vol. 36, no. 7, pp. 1307-1324, 2020, doi: 10.1007/s00371-019-01726-2.

[5] E. Hotaj, "How to Get Beautiful Results with Neural Style Transfer", *Medium*, 2020. [Online]. Available:https://towardsdatascience.com/how-to-get-beautiful-results-with-neural-style-transfer-75d0c05d6489. [Accessed: 20- May- 2021].