COMP 3359
Artificial Intelligence Applications Final Report
Group 1

Yang Yuezhi (3035603033) Pan Huijie (3035534622) Li Shanggen (3035551228)

1. Project overview

Neural style transfer, as an artificial intelligence algorithm to merge two images to generate a new image whose content is similar to one while style is similar to the other, has gained increasing attention nowadays. Three different methods developed separately by Gaty, Ulyanov, and Chan have been proposed to use the power of Convolutional Neural Networks (CNNs) to achieve such tasks, which all have produced reasonably good results.

In this project, our group reimplemented three neural style transfer models according to the original paper published by the above three researchers, which generally represented three broad categories of neural style transfer method. They are image-optimization-based methods, singular style model-optimization-based methods, and multiple styles model-optimization-based methods. Besides, we have also done a wide range of ablation experiments through the models to investigate the effect of different varieties of input and hyper-parameters turning on the performance of the model. Also, qualitative comparisons between different models are conducted, and possible limitations and error analysis are discussed. In the end, future improvement suggestions are given based on the analysis.

The detailed codes, implementation details and the elaboration of the original papers are presented in three separate Jupyter notebooks along with this report. We highly suggest readers to walk through the Jupyter notebook first before proceeding to the latter section of this report. (All the code, result and notebook are available on the GitHub https://github.com/yyuezhi/HKU-COMP-3359-Group-I)

2. Report of tasks achieved

- ### Reimplement 3 different neural style transfer models and original paper explanation

This part is in the jupyter notebooks.

- ### Hyperparameter tuning experiment

In this section, we will show some experiments conducted by our group.

Since in both image optimization and model optimization-based models, the loss function is calculated similarly. That is, reflecting the relative differences between generated images and target content and style images in several feature maps extracted by the descriptor network.

- ➔ The content loss is calculated as the norm of differences in feature maps between content and generated images.
- ➔ The style loss is calculated as the norm of difference in the gram matrix of feature maps between the generated and style images.

Therefore, it would be beneficial for all three models if we investigate the effect of turning hyperparameters in the descriptor network qualitatively and hence better fine turning our model. As a result, we decide to conduct a series of experiments on hyperparameter turning just on the image-based optimization model, whose descriptor network is VGG 19 trained from ImageNet data, and we believe these experiment results could be easily generalized to other two models. Also, due to the very long training time (5~6 hours) required by the other two methods, only performing such experiments on the image optimization-based methods is the reluctant choice.

In every round of these experiments, three to four style images representing different art styles are used to generate new images with four scenery images of HKU campus.

Content Images Figure 1

Style images Figure 2

2.1 Investigation on Style layers.

2.2 Total variation loss experiment.

2.3 Investigation on content layer.

These three parts are same with the intern protype. Hence, you can refer to that.

2.4 Investigation on different descriptor network

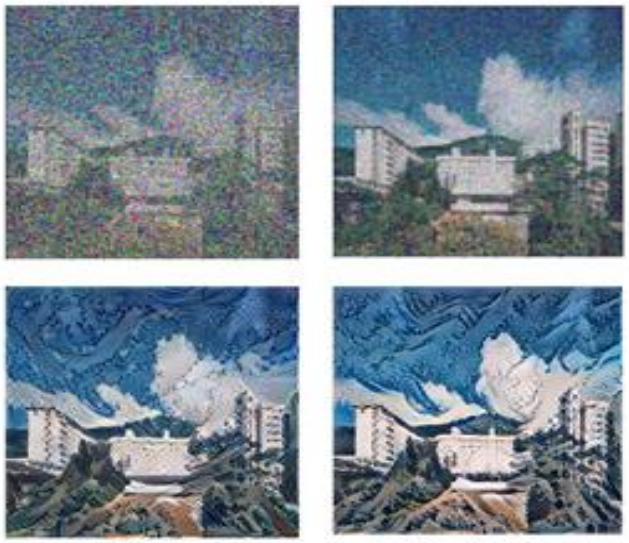
Up to now, all the experiments conducted on image-based optimization methods are on the VGG 19 network, as suggested by the Gaty's original paper. We wish to investigate the results using different networks. In this experiment, we try a descriptor network with a similar structure, like VGG-16 and those with much different structures like ResNet. In each different descriptor network, we have finetuned the hyperparameter such that the ratio of different weight components would maintain at a similar level as for other descriptor networks. For the case of choosing the content and style layers, we choose mainly the middle layers in each model, as discussed in previous experiments whose layers contain better content and style information.

Experiment on different descriptor network Fig 10

We can see from the figure that the resultant images produced by VGG 19 and VGG 16 are similar due to their similar architecture. The VGG 16 network tends to produce brighter and more vibrant colors in the images than the VGG 19 network with better quality. We think this is because VGG 16 has smaller architecture, which means its middle layers are relatively shallower than those of VGG 19, thus preserving much lower level information to contribute to color change. Besides, we note that the images produced by ResNet50 and ResNet 101 are significantly inferior to those of VGG networks, as there are more checkerboard pattern and higher frequency noise in the image. It suggests ResNet might not be a good choice for this particular task. We are not sure about the reasons for the inferior performance of ResNet. While considering the fact that ResNet 101, who has deeper architecture would have even inferior performance compared to ResNet 50, we suppose that this may due to the deeper network extract higher-level information at an earlier proportion of network hence result in such noise, which similar to the noisy pattern produce from the last deepest layers in experiment

2.5 Noise ratio experiment

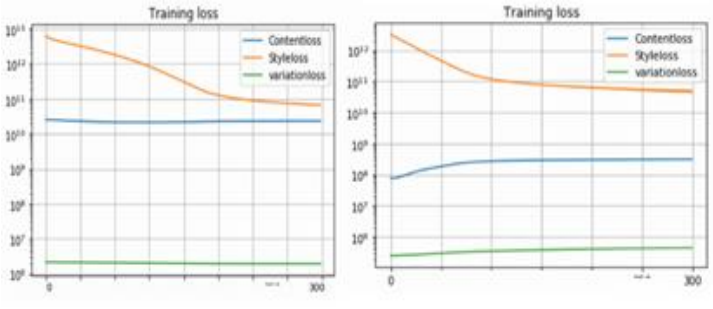
In this experiment, we try some different noise ratios when initializing the images. The images would be initialized to completely gaussian noise images when noise ratio equals to one while it would be the same as content images when noise ratio equals to 0. The higher noise ratio used, the less similar the initialized images to the content images.



Noise ratio = 0.7 Noise ratio = 0.2

Two images above are images initialized with noise ratio 0.7 and 0.2 respectively
Two images below are the resultant images generated respectively Fig 11

Qualitatively speaking, there is no significant difference between two images. Although the training loss (shown below) indicates that both images converge at the end of the training, the variation of style loss clearly indicates that the training converges earlier with low noise ratio initialization, as it takes much longer times for images with larger noise ratio to assemble content images features. Interestingly, the content loss with low noise ratio training actually goes upward during the training. This can be explained by the fact that when images are initialized similar to content, the content loss would be low. While, the training proceeds the stylized images are more different to content images compared with initialization due to the styled element added to the images. This echoes the fundamental



Noise ratio = 0.7 Noise ratio = 0.2

Training loss with noise ratio 0.2 and 0.7 respectively Figure 12

intuition of the loss function of this model; to let new generated images find a balanced point between the content and style images.

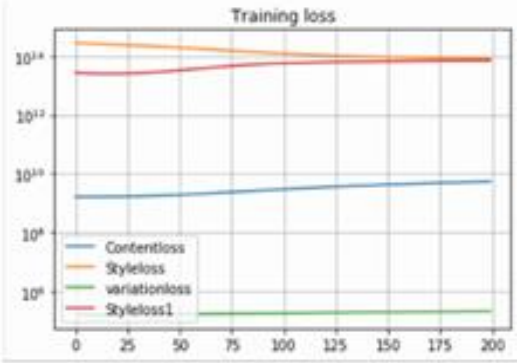
2.6 Double style experiment

We also attempted to modify the loss function so that the images can contain two styles together. We added another additional style loss term to the equation, which computed the gram matrix difference between the generated images and the second style images, as shown in the equation below.

$$J(G) = \alpha J_{content}(C,G) + \beta J_{style1}(S_1,G) + \gamma J_{total\_variation}(G) + \lambda J_{style2}(S_2,G)$$

Two style experiment reuslt Figure 13

The result is not satisfactory as the images contain too much noise and high-frequency components, which defects overall quality. Also, the images fail to exhibit two styles but predominantly exhibit one style or no style instead, even if two style loss yields similar loss value at the end of the training. Our group thinks this is because two imposing style loss constrains each other, which limits the images to evolve in neither direction. As a result, only high-frequency artifacts are shown, which is common in both styles. Also, from the fourth image and the third image, we observe that some styles, such as The Great Wave of Kanagawa style are easier to express and hence predominant other styles such as Monet impressionism style.

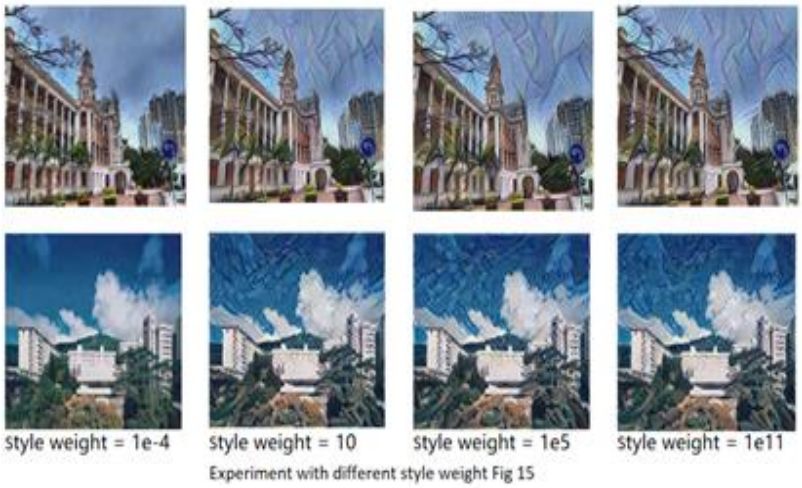


Two style experiment loss Figure 14

2.7. Style weight experiment

Our group also investigated how different style weight values will affect the generated image.

In this experiment, we set the content weight to 1e5 and the total variation weight as 100 unchanged while testing different style weights.



Experiment with different style weight Fig 15

It could be easily seen that the generated images become much more similar to the content images with low style weight of 1e-4, while there are no significant differences when the style weight increases to a much higher level, meaning the style weight hyperparameter is only sensitive in a certain region but not the whole. Even if with a high style weight of 1e11, the final style loss is much higher than content loss, the style effect would only have limited influences on the generated images.

2.8 Experiment on different resolution of images

The effect of different size styled images on the generated images are also investigated. We use the same set of style images but with two different dimensions. As for the large style image, we use the original dimension for style images, which is roughly 1024 x 1024. As for the small size style image, we resize it to 256x256.

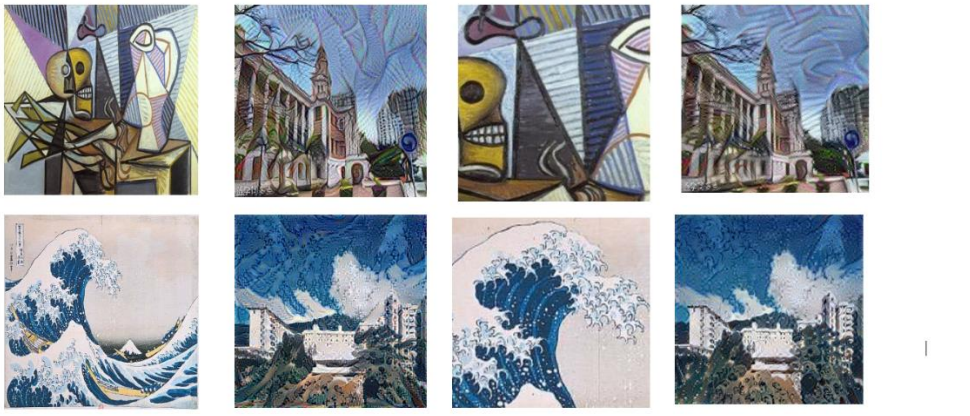
Style image size = 256 x 256 Style image size = 1024 x 1024

Experiment with different style size Fig 16

The result is quite interesting, as we can see, the generated images are quite different for different style image sizes. The images generated using style images of large size will show the brushstrokes of the style image much more. So, when the size of the style image increases, the generated image will be much more affected by the style image, and the brush strokes of the painter will also be captured much better.

2.9. Experiment with cropped style images

As the gram matrix measures the correlation of features in the style matrix. It is reasonable to investigate whether the cropped style images, which eliminate many other features in the original style images but emphasize certain features, would influence the generated images significantly. For example, for The Great Wave off Kanagawa style image, the style of generated images is mainly represented by the wave like cloud and the dot-circle pattern in the sky.



Original style images Resultant images with original style images Cropped style images Resultant images with cropped style images

Cropped style image experiment

If we feed the model with cropped style images which enlarges these features, we expect the generated images would have a more prominent style effect. Similar arguments apply for other style images.

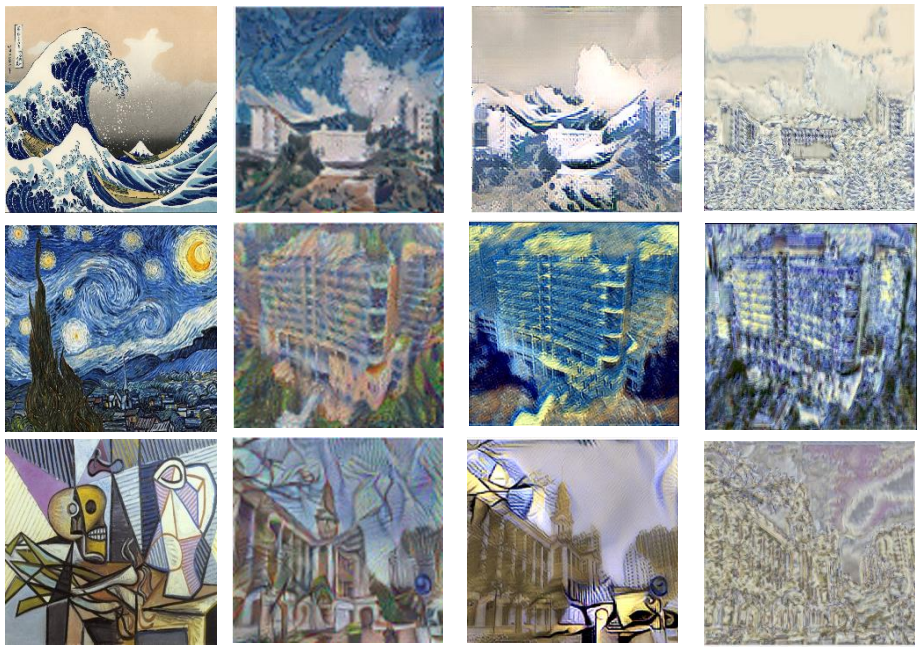
From the result, we could observe that the size of features in generated images (i.e., the color circle curve in first images and the dot-circle pattern in the second) is larger in cropped version compared to that in the original version. This could be explained by the increase in the relative size of features to the images in cropped ones compared with the original ones. Also, the cropped version shows less prominent style element to original version despite all the other parameters like style weight are set unchanged, suggesting the possible relationship of relative compactness of style in the style image to style effect of generated images.

3. Results on model evaluation and comparison

In this section we will show the output image from the three different models and then compare them based on the result.

- Result

On the left-hand side is the style image and the output images on the right-hand side are list in the following order. (Image optimization, Singular style model optimization, Multiple style model optimization)



Style image image optimization singular style model optimization. Multiple style model optimization

- Comparison

3.1 Based on optimization algorithm:

In Jing's paper, these methods have been divided into three different categories. However, essentially the first one is doing the optimization based on the content image.

And the remaining two is using the CNN layers to store the style image information, which is belong to the feed-forward network methods. You can find more detailed algorithm explanation in the Jupyter notebook.

3.2 Based on generated images:

As we can see from the result above, the Image optimization method pays more attention to study the style image's structures and line elements. The content image color is still largely preserved on the output images. However, since the other two methods generate the style and structures of content images simultaneously, so they produce inferior results compared to the first method, in terms of color, coherence of detail structures, as well as the harmony of merging between content and style element. For the second method, the result generated tends to have an uneven distribution of style element across the images and many single-color blocks across, which deteriorates its visual quality. Furthermore, the last method could only and remain the content image's structure and use the style image pattern to reconstruct the main shape of the content structure. We believe this difference is the result of different style image study strategies talked above.

3.3 Based on time:

We us a table to show the learning and generating time's difference.

		Image optimization	Singular style model optimization	Multiple style model optimization
Learning	Style in a net	One	One	Multiple
	Times per style per step (Batchsize=4)	0.9 second	1.0 second	20 second
	Total training time	4 mins	0.69 hour	5.5 hours
	Incremental enable	No	No	Yes
Generating	Time per image (GTX1080Ti,256*256)	60 second	0.31 second	0.21 second

4. Possible limitations and remaining issues

In the comparison part, you may find our results are not so appealing. So, in the following section, we will talk about our reimplement models' limitation and how we manage to solve these problems model by model.

4.1 Image optimization-based method

The output image this model generate is similar to the original paper. So, we only discuss the limitation about the model itself.

Firstly, producing a styled image requires a fresh training on noise images every round, which is time consuming compared to the other two models. Secondly, this model does not perform well enough in preserving the coherence of fine structures and details during the training since CNN deep level features maps inevitably lose some low-level information. Moreover, the variations of brush strokes and the semantics and depth information contained in the content image are not reflected in the loss function hence not present in the generated images, which are important factors in evaluating the visual quality.

4.2 Singular style Model optimization-based method

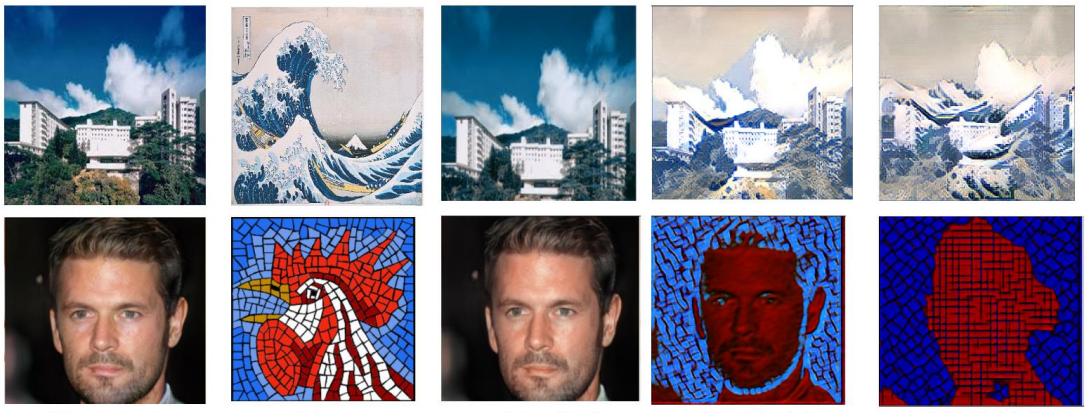
Since this model design basically follows the loss function of image optimization-based model, which makes them suffer from the same issues as the previous one, such as lack of coherence of details and ignorance of depth information. Although the advantages of forward CNN feed generation greatly reduce the time required to produce one single image, and achieve real-time style transfer, each trained network could still only to provide one style generation. It may limit its potential to be developed into practice.

What's more, it is clear from the comparison section that this model does not produce a good result. We conclude that it suffers from the following issue. Firstly, the style representation in the images is not prominent, and the result is too similar to the content images. Secondly, the degree of style is not uniformly distributed around the images; some parts of the images have a prominent style element while the others do not. In addition, there seem to be many single-color blocks in the images, which deteriorate the visual effect of the images.

These issues still could not be properly addressed before the final work submission, and it clearly does occur in the original paper. With the help of model.summary() function in TensorFlow and using the exact loss function calculation method in the image-based model, we can guarantee that there is not an explicit bug in the algorithm implemented. Therefore, we conclude that problems could only occur because of inadequate training methodology or hyper-parameter turning. Below presents the effort we take to solve such a tangled problem.

4.2.1. Investigation on different style weight and content weight

Since the style and content weight only determines the relative ratio of style and content loss, we fix the content weight to 1 and only change the style weight. Below is the result produced when training the model using one single style and content images with learning rate of 1e-3. Training steps is set to be 2500, which is enough to converge.



Model-optimization based tuning experiment

From the above result, we could observe that the model performs satisfactorily well to produce the original content images when the style weight equals to zeros, despite some color distortion due to the inevitable loss of low-level information in CNN models. While when the style weight gradually increases, we see growing influences of the style element in the generated images which distort the content images. It does not seem like the style element, and content images merge coherently in the output images. Instead, we could see that many regions have an abnormal amount of pure (Red, green, blue) color, which distorts the aesthetic effect for images with a style weight of 1e-6 and 1e-3. We also experiment with a different style of weight. The result shows that the images do not have significant changes with style weight bigger than 1e-3 and smaller than 1e-6. Besides, we find out that model trained on each specific style requires extra fine turning on the weight of loss term, as an example from here, the relative better result of the first images come from the when style weight equals 1e-3 while the second images perform between when style weight equals 1e-6.

4.2.2 Investigation on the initialization method

We also find that different initialization method also has many influences on the images generated. As suggested by the original paper, the weight of the model should be initialized by the Xavier's method (i.e. Glorot normal initialization), the result produced

is not satisfactory. Therefore, we also try He's initialization method, which is considered as a better initialization method for models using LeRU families as activation function.

Xavier's initilization He's initilization
Different intilization on model based optimization with style weight 1e-6

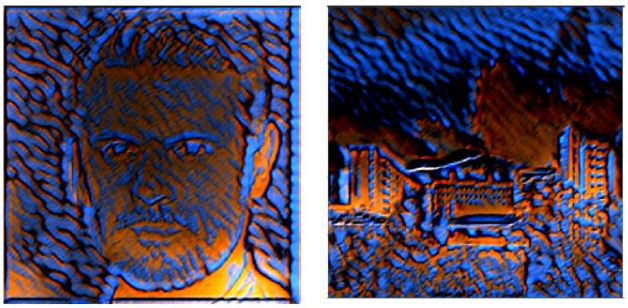
The result shows that different initialization methods do have a considerable influence on the generated images, but overall the Xavier's initialization method do yields better result.

4.2.3 Investigation on different model architecture

By tf.summary() function, our group finds out that this model has a total of 75269 trainable parameters, which is quite low. So, we suspect that probably the model is not flexible enough to learn all the content and style information adequately, especially given that we repeatedly use only a few content images to train the network instead of using COCO dataset as the original paper suggested due to computational resources constrain. Thus, we decide to use another singular style model-based optimization method suggested by Justin Johnson in the paper titled "Perceptual Losses for Real-Time Style Transfer and Super-Resolution", to see whether the similar problem also occur there. The detailed architecture of the model is given below, together with the result generated after fine tuning the hyper-parameters. The implementation code is denoted in the bottom of the notebook "singular style model optimization-based method" with no other markdown cell explanation.

Layer	Activation size
Input	$3 \times 256 \times 256$
$32 \times 9 \times 9$ conv, stride 1	$32 \times 256 \times 256$
$64 \times 3 \times 3$ conv, stride 2	$64 \times 128 \times 128$
$128 \times 3 \times 3$ conv, stride 2	$128 \times 64 \times 64$
Residual block, 128 filters	$128 \times 64 \times 64$
Residual block, 128 filters	$128 \times 64 \times 64$
Residual block, 128 filters	$128 \times 64 \times 64$
Residual block, 128 filters	$128 \times 64 \times 64$
Residual block, 128 filters	$128 \times 64 \times 64$
$64 \times 3 \times 3$ conv, stride 1/2	$64 \times 128 \times 128$
$32 \times 3 \times 3$ conv, stride 1/2	$32 \times 256 \times 256$
$3 \times 9 \times 9$ conv, stride 1	$3 \times 256 \times 256$

Architecture for Johnson's model see detail at the Supplementary Material of this paper



Result generated by johnason's model using same content and style image

The Johnson's model also shows the same problem of inadequate mixture even though it is much more flexible than the original model in terms of the number of trainable

variables. This indicates that such problem could be model independent, which also occur in different architectures.

4.2.4 Investigation on different training methods

Our group eventually comes up with a method which could produce a slightly better result compared with that with a naive training method, which directly trains the network from scratch with fixed learning rate. The new methods would firstly train with style weight settled to zeros to allow the network to produce an image highly similar to the content images in a relatively high learning rate. Then, we would train with desired style weight with learning rate decay to half of it every 1250 steps. This allows the model to "make modification" on an image highly similar to the content images so that the problem of content information cannot be adequately shown in the generated images could be addressed.

4.2.5 Conclusion

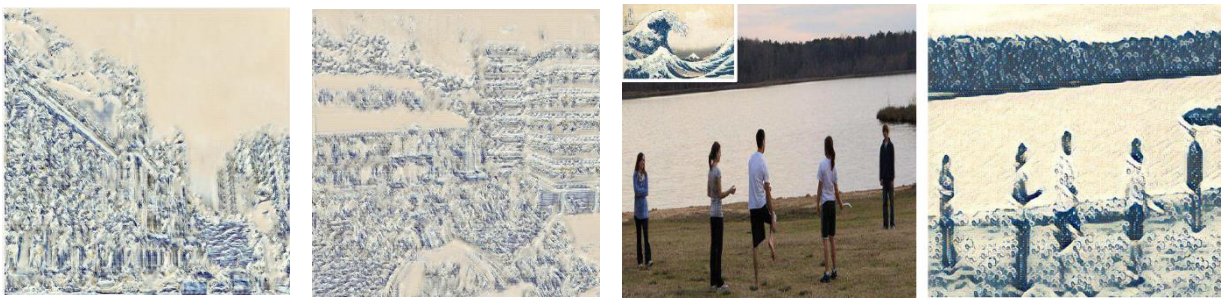
In conclusion, from the above-mentioned experiment, our group partially solves the problem and manage to improve the result by adopting a new training method and fine turning hyper parameter. We also find that the model architecture and weight initialization have somewhat effect on this problem while it is not significant. Due to the constrains in computational resource and limited project timing schedule, we are not able to conduct further experiment to solve this problem thoroughly. While our group believes that the general direction to solve this problem should be to find a more effective training methodology and using a more robust set of hyper parameters.

4.3 Multiple style model optimization

This model should be the best because it enjoys the least image generating time and ability of incremental Training. However, like the previous model, our implementation 's transfer result is also not that perfect. We conclude that the reasons could be

1. Input image size: instead of using the (512*512*3) as paper suggested, we use (224*224*3) because of the limited GPU memory size and too many computing hours.
2. The gram matrix didn't work so well. Because the gram matrix is used to compute the relationship between different features. From the result of the output image we can see the styled pattern seem not go to the right place, especially in the last Picasso's style.
3. Limited training epoch: Suggest training epoch is 300k, however we don't have too much time to do these amounts of training. The second problem may be solved with more training epoch.

Considering our limited GPU resource and limited time, we have to conduct our experiment just in one style image first and use the suggested input image size. (you can find the different models in our Jupyter notebooks)
Here are the results of large input size and 1k literation. We also notice that after 1k literation the output image and style loss do not have too much change no matter how we decay the learning rate.



Compared to the previous output image, we can see that the style pattern e.g. (color and the wave carve) emerges more coherent and the output image is clearer because of change the input image size to (512*512). However, compare to ideal output in the original paper, it is still not satisfying. The gram matrix's problem seems still exist because the dark color area still has too much green-white style pattern.
Because of the time reason and the large amount of time required to train this model, we could not carry out more experiment on this model. But we may conclude that the reason is mainly because of the limited training epoch.

4.4 Future improvement direction of all three models

Overall, based on the paper we reviewed in these days, we find all these methods are based on the fitting of the overall features of the style image but ignore the corresponding relationship between the pixels of the images. This may result that the final transfer result is consistent with the overall style image but with some local errors (such as the pure color block). In addition, we also notice the best transfer results often happen when the style image and the content image have something similar i.e., the same amount of color, structure. Transferring a totally different content and style image may also be a limitation to these three methods.

However, in the recent studied paper, we find that by using the framework of Image Analogy and the high-level feature extracted by a deep neural network (DNN), researchers from Microsoft Research Asia can fully establish semantic matching between images, which somehow could solve the problem mentioned above. In addition, there are many methods controlling perceptual factors in neural style transfer, such as color

control and space control, which can also be used in our three methods to improve performance.

5 Conclusion

Although we do not get the perfect transfer results in our last two reimplementation models, we do manage to improve the result as much as we can in a limited time. Future improvement could still be made, such as increasing the training time, adapting the exiting controlling method to fix our problems. However, we have a better understanding of the AI application and learn a lot about the Neural style transfer through this project. We hope to do more further study in the future.