Style Transfer Showdown: A Comparative Study

Ryan Yih-Yun Liu

Abstract

In recent years, style transfer applications have increased significantly that allow users to generate images with the content image and various style targets chosen by them. This report provides a comprehensive overview of the evolution of style transfer in the image processing field which analyzes and evaluates three different methods of style transfer, including two from neural style transfer and one from a CLIP-based model, while also highlighting the benefits, limitations, and potential use cases of each approach. Furthermore, the report also thoroughly examines the development of style transfer, from its origins in computer vision to its present position as a flexible and effective tool for producing creative material. The report concludes by offering the readers a guideline on how to choose the most suitable methods based on their requirements to get the desired style transferring results.

1. Introduction

Non-parametric algorithms for synthesizing texture from given input have been available for decades [1,2]. Lee et al. later improved these algorithms by incorporating edge information. Although these previous works produced impressive results, the emergence of convolutional neural networks enabled people to inspect high-level features instead of low-level ones such as edges and brightness.

Zeiler and Fergus demonstrated the effectiveness of convolutional networks in representing images [3]. Leon et al. built on this work and showed that style and content representation can be decoupled in convolutional neural networks [4]. Building on this, Leon et al. successfully synthesized a style image onto a content image, which became the first version of neural style transfer (NST) [5]. Following the success of single style transfer, blending two styles into the content image was made possible by adjusting the ratio of the two style losses.

However, the NST method requires time for training in order to optimize the loss between the target image, the input content, and the style image. In contrast, Justin Johnson et al. proposed another method [18], which pre-trains a network on a specific style image. As a result, the style transfer process can be performed in real time by simply feedforwarding the input image through the pre-trained network.

However, both NST methods require the user to provide a style image as input, which can be impractical in some cases. For example, the user may want to synthesize a specific artist's style without knowing the name of the artwork or may only have a general idea of how to synthesize a certain style without knowing which specific style image to use as a reference.

The emergence of Contrastive Language-Image Pre-training (CLIP) [19] has addressed the limitations of NST. CLIP is a model that learns joint representations of images and text that are trained on a large dataset of image and text pairs. It allows the user to input any text prompt and delivers the semantic textures of the text conditions. Gihyun Kwon et al. [20] use a lightweight CNN to make the content image follow the text condition by matching the similarity between the CLIP model output of the transferred image and the text condition.

There are extensive experiments on original NST [5], feedforward NST [18], and a CLIP-based method [20] in the following report.

2. Related work

2.1 VGG-Networks

The history of convolutional neural networks can be traced from a classification challenge started from 2010 called "ImageNet large scale visual recognition challenge" (ILSVRC), where in 2012, the first convolutional neural networks-based winner came out called AlexNet [6]. The AlexNet has 8 layers (5 convolutional layers and 3 fully connected layers), which outperform other traditional machine learning methods at the time.

Moving on to 2014, a new architecture known as VGG-Network (Visual Geometry Group) provided by Simonyan & Zisserman [7] win the ILSVRC 2014. The VGG-Network improves AlexNet by a team from Oxford, the VGG19 is a variant of the VGG-Network, which consists of 16 convolution layers, 3 Fully connected layers, 5 MaxPool layers, and 1 SoftMax layer.

Although after 2014, there are some convolutional neural networks came out, such as Residual Networks (ResNet) [8], and Squeeze-and-Excitation Networks

(SENet) [9]. It can be imagined that these latest CNNs can have a better result in the neural style transfer task. However, for the experiments of the original NST in this report, I will focus on the VGG19 network, which is the same convolutional neural network used in the original NST paper [5].
Figure 2 shows the architecture of VGG19, source from [13].

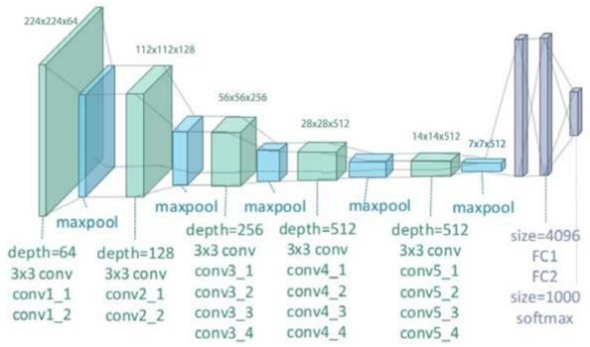


Figure 2. Visualization of VGG19's architecture

2.2 Image Representation in Convolutional Neural Networks

The core idea of neural style transfer is to utilize image representation in convolutional neural networks. Zeiler and Fergus [3] demonstrated this by analyzing the same pre-trained network used in neural style transfer [5], which was trained on ImageNet. They found that the low-level layers (layer1, layer2, layer3) in the feature maps tend to represent the texture of the image, such as edges and simple textures, while the high-level layers (layer4, layer5) tend to represent semantic features, such as object parts and complex textures. Figure 1 depicts neural style transfer results by extracting from content layer relu4_2 (2nd ReLU function in layer 4) and relu2_1 (1st ReLU function in the VGG-19) with Van Gogh's Starry Night and Virginia Tech's building as style and content inputs, respectively.

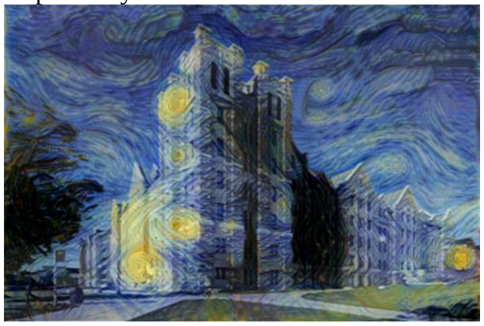
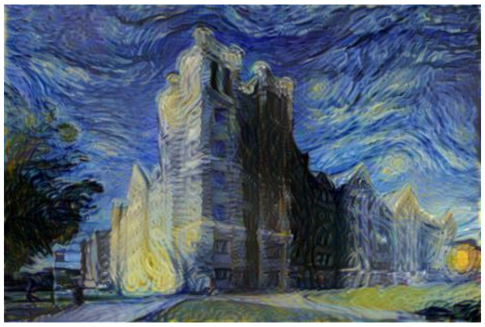


Figure 1 (a) Extracting content from layer relu2_1


(b) Extracting content from layer relu4_2
Image (a) and (b) have the exact setup except for the choice of the content layer, where we can see image (b) preserve more semantic information than the image (a).

Understanding how the convolutional neural networks perform can help us target the specific layer to do the style transfer while preserving the semantic information of the content image. In this report, I use the same convolutional neural networks as in the original neural style transfer paper [5], the VGG-Network pre-trained on the ImageNet dataset for object recognition and localization. Therefore, by combining these two papers [3,5], we can have a understanding of the idea of neural style transfer.

2.3 Optimizer: L-BFGS v.s. Adam

PyTorch is an open-source machine learning framework used for the GitHub repositories [10, 20, 21, 22] I reference in this report, there are two different optimizers that can be applied to the experiment of neural style transfer to optimize the loss. One is the Limited-memory BFGS (L-BFGS) from [11], according to its Wikipedia definition, L-BFGS is an optimization algorithm in the family of quasi-Newton methods, it uses a Hessian approximation to estimate the parameter space's curvature. Another optimizer is Adam from [12], in place of the conventional stochastic gradient descent method, Adam is an optimization technique that may be used to iteratively update network weights depending on training data. The author shows that the L-BFGS typically produces better results than the Adam but consumes more memory. However, the result with the L-BFGS optimizer has a better style blending than the result with the Adam optimizer. Figure 3 shows the results from L-BFGS and Adam optimizer.

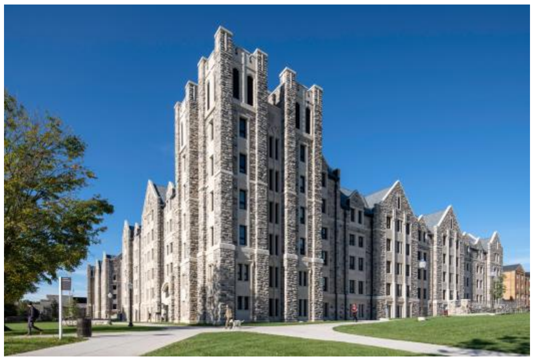
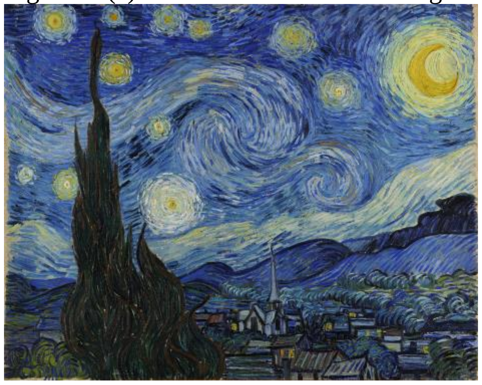
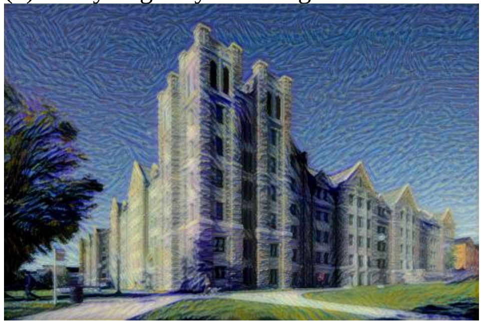
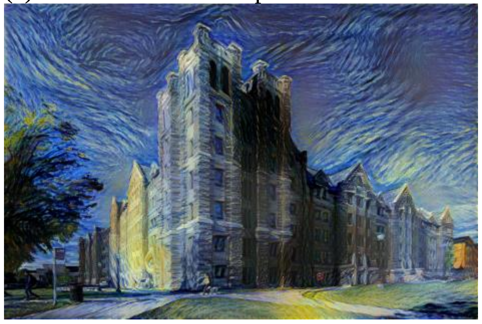


Figure 3. (a) The Pearson Hall from Virginia Tech

(b) Starry Night by Van Gogh

(c) Result with Adam optimizer

(d) Result with L-BFGS optimizer
Image (c) and image (d) have the same setup for (Architecture/ numbers of iterations/ weight of content to style/ content layer/ style layer/ initialize image/ output image size) except for the optimizer choice. (VGG19/ 1000 iteration/ 1:1000/ content layer extract from relu4_2/ style layer extract from relu1_1, relu2_1, relu3_1, relu4_1, relu5_1/ content image/ 1024).

2.4 Total Variation Loss

In image processing tasks, it is common for the output images to have undesired noise or blurring at the edges, which can reduce their overall quality and visual appeal. To address this issue, the Total Variation (TV) loss proposed by Rudin, Leonid I. et al. in [31] is often used as a regularization term in combination with other loss functions. The TV loss function, in equation (1), is defined as the sum of the absolute differences between adjacent pixels in the image.

TV loss:

$$\sum_{i,j}[\,|I(i+1,j)-I(i,j)|+|I(i,j+1)-I(i,j)|] \quad (1)$$

The TV loss function (1) considers only adjacent pixels in the horizontal and vertical directions because these pixel pairs are the most significant in terms of capturing variations in pixel values that affect the overall smoothness of the image. By focusing on these pairs, the TV loss function can effectively penalize large variations in pixel values across adjacent pixels that are indicative of image noise or abrupt changes in image intensity. The TV loss encourages smoothness and edge preservation in the output images by penalizing large variations in pixel values across adjacent pixels. This helps to reduce noise, preserve edges and boundaries, and sharpen blurry images. As a result, the use of TV loss can significantly improve the quality of output images in image processing tasks like denoising, deblurring, and super-resolution.

2.5 Feed-forward image transformation and Perceptual Loss

To overcome the training time required by the original NST [5], Justin Johnson et al. proposed a feed-forward image style transfer technique by pre-training a feed-forward convolutional neural network with perceptual loss functions to optimize the output and ground-truth image in an efficient way since it requires only a forward pass through the pre-trained network [18]. Furthermore, unlike other image transformation tasks, which only consider the per-pixel loss functions such as super-resolution [23], colorization [24], and segmentation [25], the perceptual loss in [18] refers to the loss function in the original NST [5]. The reason is that simply using the per-pixel loss function would not produce a visually pleasing result as it tends to prioritize pixel values and color information rather than capturing the overall style and texture of the input style image. On the other hand, using perceptual loss functions will take into account higher-level features by minimizing the difference between high-level image feature maps of the input and output image from a pre-trained CNN, e.g., VGG19, which can lead to a better style transfer result.

2.7 CLIP (Contrastive Language-Image Pre-Training)

CLIP is a neural network model developed by OpenAI [26] for multimodal learning, trained on 400 million text-image pairs. The model uses a contrastive learning approach, learning to associate text and image pairs that belong together while differentiating between pairs that do not belong together.

StyleCLIP [27] successfully controls the generation process by exploring the learned latent space of StyleGAN [28] with the state-of-the-art performance of CLIP. However, StyleCLIP manipulates images within the trained domain during latent exploration. To address this, StyleGAN-NADA [29] proposes a modification method that only uses text conditions without additional training images.

Although these methods can manipulate images with text conditions, they depend heavily on pre-trained generative models, and the generated images are confined to trained image domains. Therefore, Gihyun Kwon et al. [20] propose a new model that can transfer the texture of text conditions to a content image regardless of the image domain.

3. Methods

Overview

In this report, I explored three different style transfer methods: the original Neural Style Transfer (NST) [5], the feedforward NST [18], and the CLIP-based method [20]. For each method, I performed a series of experiments to investigate the effectiveness of different settings and approaches.

For the two NST methods, I did experiments on content and style trade-off, initialization image, and total variation loss on the original NST. Since the idea of the feedforward NST method is speeding up the style transferring time from the original NST, I will compare the style transfer results from both NST methods.

For the CLIP-based method, I experimented with different numbers of cropped patches, and different text-content image pairs.

Finally, to compare three style transfer methods with the same style targets, for the CLIP-based method which only takes text conditions as input, I chose the text conditions that can refer to well-known paintings or an object so the corresponding style image can be applied to the first two NST methods as well.

3.1 Double Style Transfer

Before getting into double style transfer, it is necessary to understand how to transfer a single style to the content image first. From the original neural style transfer paper [5], we know how content representation and style representation works, also how to synthesize a style transfer image.

3.1.1 Content Representation

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

In equation (2), $\vec{p}\ and\ \vec{x}$ are the original content image and the generated image, where $P^{l}\ and\ F^{l}$ are the respective feature representation in layer l. By defining the squared-error loss between the two features' representation as in equation (2), we can minimize the loss between them to make the generated image’s content representation close to the original content image to reconstruct the content information in the final synthesized result.

3.1.2 Style Representation

$$G_{ij}^{l} = \sum_{k} F_{ik}^{l} F_{jk}^{l} \qquad (3)$$

The G in equation (3) is given by the Gram matrix, where G_{ij}^{l} is the inner product between the vectorized feature maps i and j in layer l.

$$E_{l} = \frac{1}{4N_{l}^{2}M_{l}^{2}}\sum_{i,j}(G_{ij}^{l} - A_{ij}^{l})^{2} \qquad (4)$$

$$L_{style}(\vec{a},\vec{x}) = \sum_{l=0}^{L} \omega_{l}\, E_{l} \qquad (5)$$

In equation (5), $\vec{a}\ and\ \vec{x}$ are the original style image and the generated image, where the ω_{l} is the weighting factors of the contribution layer l to the total loss.

3.1.3 Single Style Transfer

$$L_{total}(\vec{p},\vec{a},\vec{x}) = \alpha L_{content}(\vec{p},\vec{x}) + \beta L_{style}(\vec{a},\vec{x}) \qquad (6)$$

In equation (6), we can jointly minimize the total loss which is the distance of the feature representation between an initialized mage (the original paper initializes with a white noise image, there will be another initialization setting as follows). Finally, the $\alpha\ and\ \beta$ are the weighting factors for content and style reconstruction.

3.1.4 Double Style Transfer

After understanding how the style transfer works, we can do a little modification on equation (6) to achieve double style transfer, see equation (7) [14].

$$L_{total}(\vec{p},\vec{a},\vec{x}) = \alpha L_{content}(\vec{p},\vec{x}) + \beta(\gamma L_{style}(\vec{a_1},\vec{x}) + (1-\gamma)(\gamma L_{style}(\vec{a_2},\vec{x})) \quad (7)$$

By adjusting the γ, we can get the different combinations of the input styles.

3.2 CLIP-based Style Transfer

3.2.1 Framework

Gihyun Kwon et al. in [20] try to transfer the semantic style of the input target text to the content image, which is different from the NST methods [5,18] that the user does not have to provide a style image for reference. To address the style transfer goal, the key ideas are understanding how to extract the semantic texture from the CLIP model then apply to the content image, and how to make the quality of the output image stable.

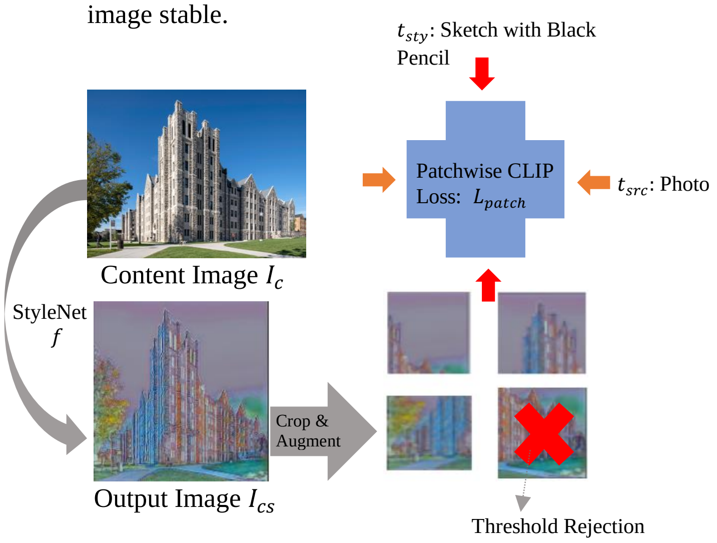


Figure 4. The schematics of the patch-wise CLIP loss in [20], which used to optimize the neural network f

The neural network f is a CNN encoder-decoder model that can capture the visual features of the content image in a hierarchical manner and stylize the image in deep feature space.

3.2.2 Different CLIP Losses

In order to guide the content image to align with the semantics of the target text, a straightforward method is to minimize the global CLIP loss function [30]:

$$L_{global} = D_{CLIP}(f(I_c), t_{sty}) \quad (8)$$

the D_{CLIP} is the CLIP-space cosine distance; however, if solely using the global CLIP loss, usually the output quality is corrupted. Therefore, StyleGAN-NADA [29] proposed a directional CLIP loss that ensures the CLIP-space direction of the text-image pairs in the source and output are in alignment, which can be formulated as:

$$\Delta T = E_T(t_{sty}) - E_T(t_{src})$$

$$\Delta I = E_I(f(I_c)) - E_I(I_c)$$

$$L_{dir} = 1 - \frac{\Delta I \cdot \Delta T}{|\Delta I||\Delta T|} \quad (9)$$

where E_I and E_T are the image and text encoders of CLIP, and the t_{sty} and t_{src} are the semantic text of the style target and the input image (The t_{src} for natural images will be set as "Photo").

Although the directional CLIP loss L_{dir} provides good results, it may decrease the quality of the output image if we only use the L_{dir}. It only encourages the output image to have similar CLIP space directions to the source image, without considering the content and style of the input image. This may result in a final image that has similar directional features to the source image but may not have the desired content and style transfer.

Thus, Gihyun Kwon et al. proposed a PatchCLIP loss [20] that is minimizing CLIP loss functions with group of patches extracted from I_{cs} to deliver the spatially invariant information, this is inspired by the original NST [5]. They crop multiple patches from I_{cs} and apply the perspective augmentation to each one before computing the directional CLIP loss, which helps the network capture more diverse textures, and makes the reconstruction of the textures become more 3D-like, see equation (10).

3.2.3 Threshold rejection

To avoid over-stylization, there is a threshold rejection in the PatchCLIP loss in [20]. With the given threshold τ, some patches that are easier than others to minimize the loss will be nullified. The Patchwise CLIP loss is defined as:

$$\Delta T = E_T(t_{sty}) - E_T(t_{src})$$

$$\Delta I = E_I(aug(\hat{I}^i_{cs})) - E_I(I_c)$$

$$l^i_{patch} = 1 - \frac{\Delta I \cdot \Delta T}{|\Delta I||\Delta T|}$$

$$L_{patch} = \frac{1}{N}\sum_{i}^{N} R(\, l_{patch}^{i}, \tau)$$

$$R(s,\tau) = \begin{cases} 0, & if\ s \le \tau \\ s, & otherwise \end{cases} \quad (10)$$

where the l_{patch}^{i} is the i-th cropped patch from the output image. Finally, the total loss function is formulated as:

$$L_{total} = \lambda_d L_{dir} + \lambda_p L_{patch} + \lambda_c L_c + \lambda_{tv} L_{tv} \quad (11)$$

where the L_c indicates the content loss, calculating the mean-square error between the features of content and output image extracted from the pre-trained VGG19 networks like in the original NST [5].

4. Results

In this section, I will show the experimental results from the original NST [5], the feedforward NST [18], and the CLIP-based method [20] respectively with a series of experiments. Then I will provide my perspective and guideline for each method's usage with quantitative analysis at the end.

4.1 Original NST

There are three different experiments in the original NST experiments: content and style trade-off, initialization image, and total variation loss. The content image will be the Pearson Hall from Virginia Tech, see Figure 3. (a), and the style image will be the Great Wave by Hokusai, see Figure 5. (a). For all the experiments, the parameters will be the same except the one been experimented with.

4.1.1 Content and Style Trade-off

From the original NST [5], the best ratio of the content weight to the style weight is 1e5 to 3e4, see Figure 5(d). To show how different pairs of content weight to style weight will affect the result, I changed the content weight to 1e7 and 1e6 while fixing the style weight to 3e4. Figure 5 shows the results.

Apparently, we can see as the content weight decrease, there are more style textures being blended on the result.

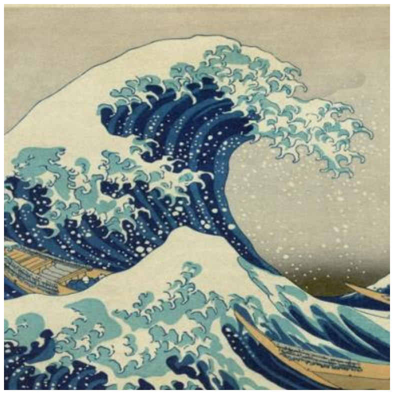
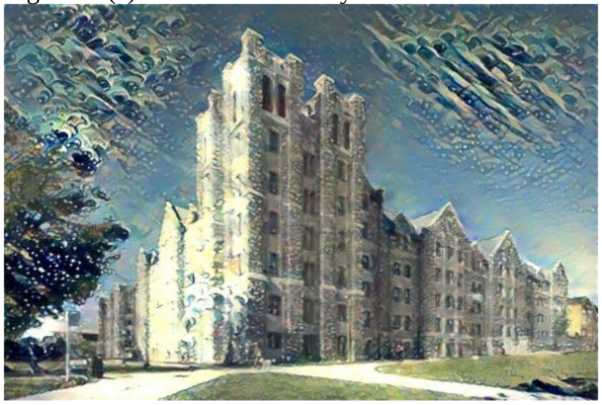
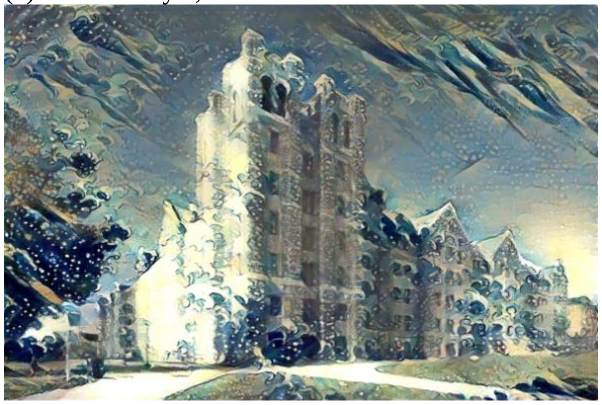
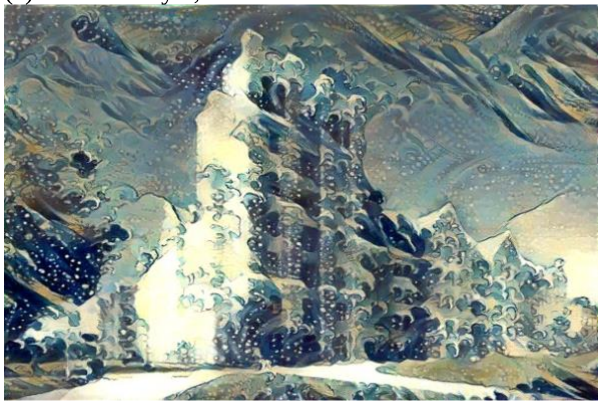


Figure 5. (a) The Great Wave by Hokusai


(b) Content: Style, 1e7: 3e4


(c) Content: Style, 1e6: 3e4


(d) Content: Style, 1e5: 3e4

4.1.2 Initialization Image

Different initialization images can have a significant impact on the results, specifically in the original NST [5] where the initialization image serves as the starting point for the optimization process to match the input content and style images. There are four different initialization methods in this experiment: the content image, the style image, and the random noise images which are the Gaussian noise image, and the white noise image, respectively.

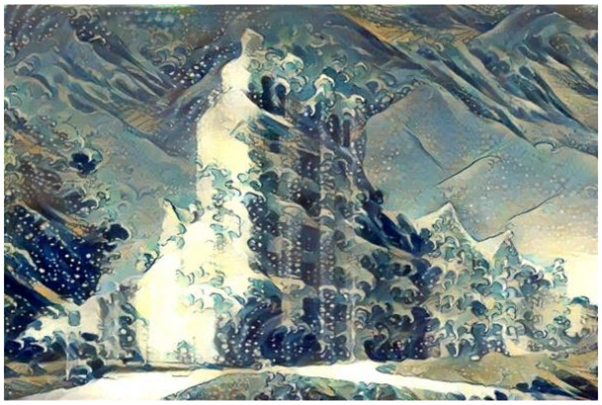
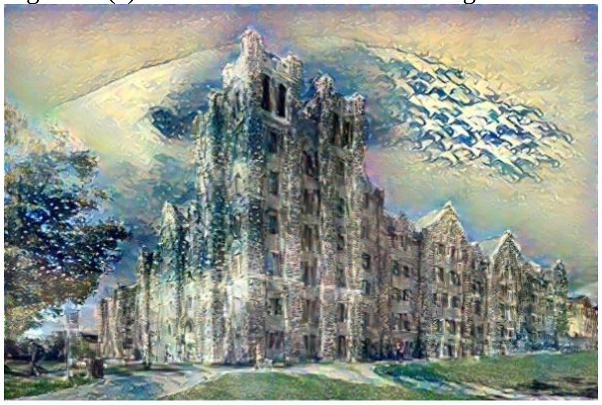
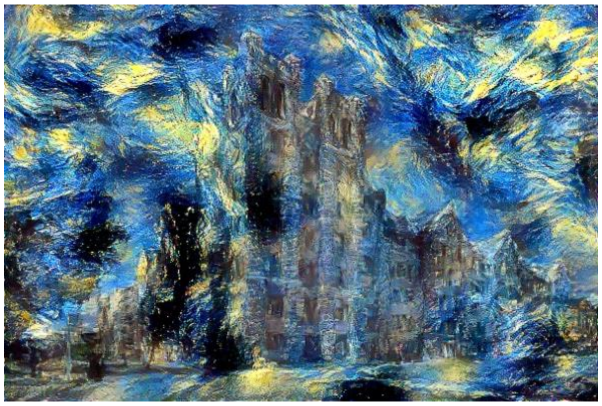


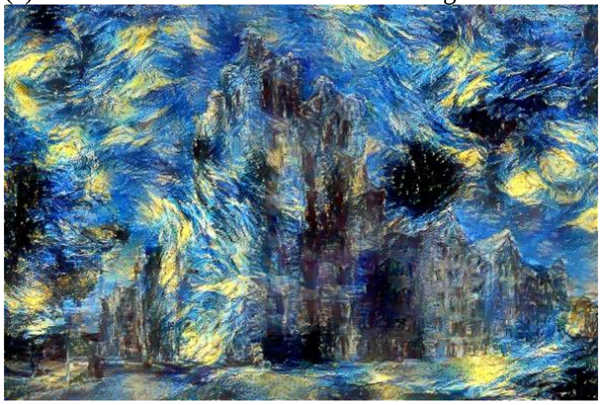
Figure 6. (a) Initialize with the content image


(b) Initialize with the style image

As we can see in Figure 6. (a) and (b), apparently result (a) tends to transfer the style to the object in the content image which is the building, and result (b) has a wave in the background which indicates the result tends to have more object from the style image.

However, for the random noise image, usually, there are two methods to choose which are the Gaussian noise image and the white noise image. The main difference between these two types of noise images is the Gaussian noise image follows a normal distribution which is helpful for capturing the overall tone and style of the style image; the white noise image contains an equal amount of energy at all frequencies which is a good starting point for capturing the overall style. To show the difference, I choose Figure 3. (b) Van Gogh's Starry Night as the style image which has a more obvious texture such as the brushstroke than the Great Wave. We can see Figure 6. (d) shows more textures than (c).


(c) Initialize with the Gaussian noise image


(d) Initialize with the white noise image

4.1.3 Total Variation Loss

We have already known that the higher the total variation loss is, the smoother the result will be. Therefore, in Figure 7., we can see the difference in the results as the tv loss increased by 10 times from Figure 7. (a) to (c), the resulting image becomes smoother.

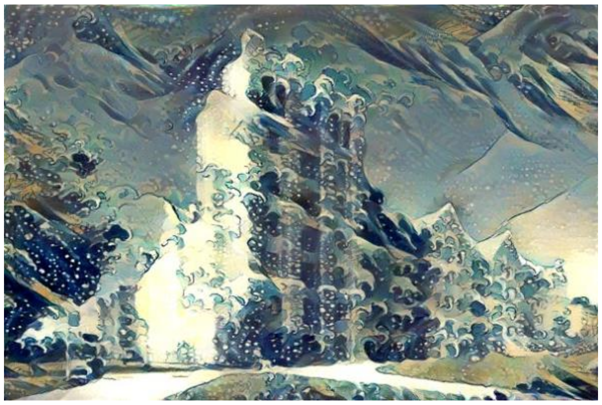
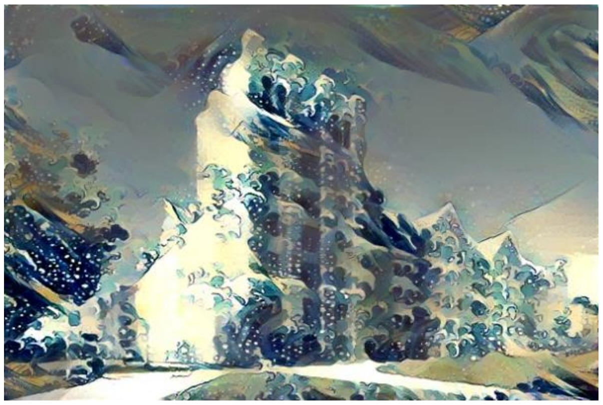
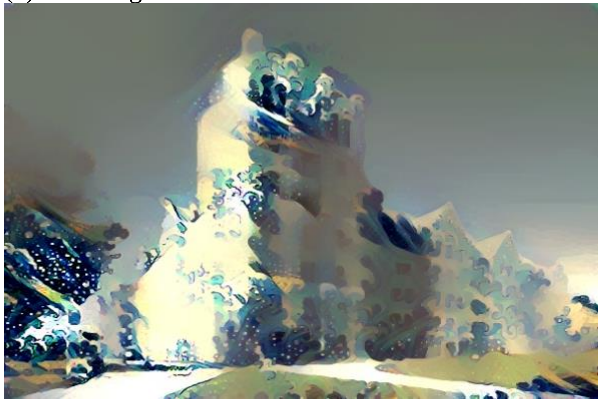


Figure 7. (a) The weight of the TV Loss is 1e1


(b) The weight of the TV Loss is 1e2


(c) The weight of the TV Loss is 1e3

4.2 Feedforward NST

4.2.1 Different Types of Style

In the feedforward NST experiments, I will show the style transfer results from several different types of style images to the content image, Pearson Hall from Virginia Tech, see Figure 3. (a). The style images contain abstract art (Weeping Women by Pablo Picasso), woodcut prints (Saint Anthony Before a City by Albrecht Dürer), and objects (Green Crystal), see Figures 8, 9, and 10 respectively, for each Figure, (a) is the style image and (b) is the result.

Although in the previous section **2.3**, we know that training with L-BFGS optimizer has a better style transfer result than with Adam optimizer but needs more computational resources and memory. The experiments in this project were done on Google Colab with limited computational resources; therefore, I used the Adam optimizer in the feedforward NST experiment.

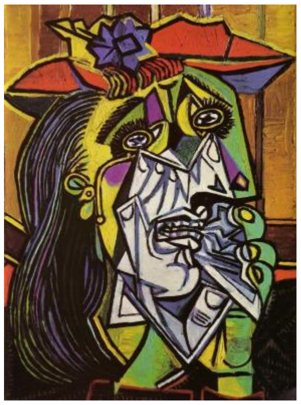
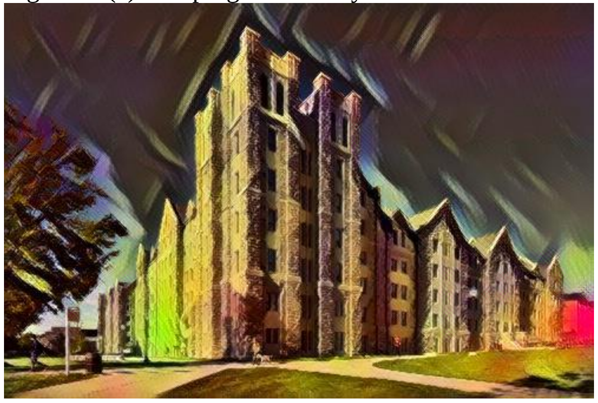


Figure 8. (a) Weeping Women by Pablo Picasso


(b)

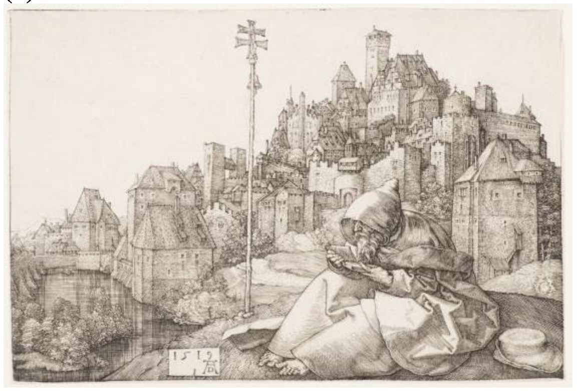
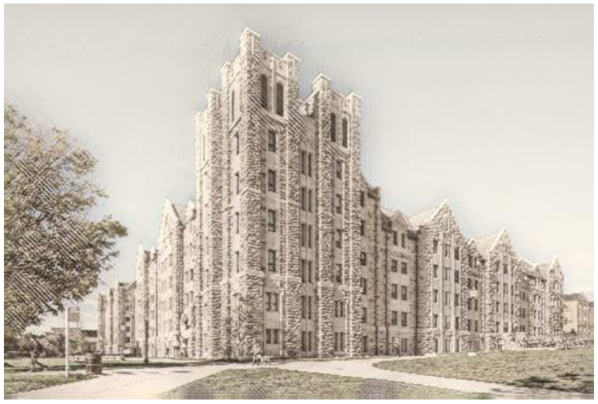


Figure 9. (a) Saint Anthony Before a City by Albrecht Dürer


(b)

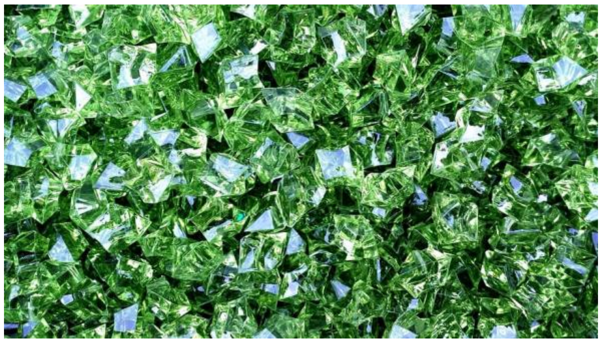
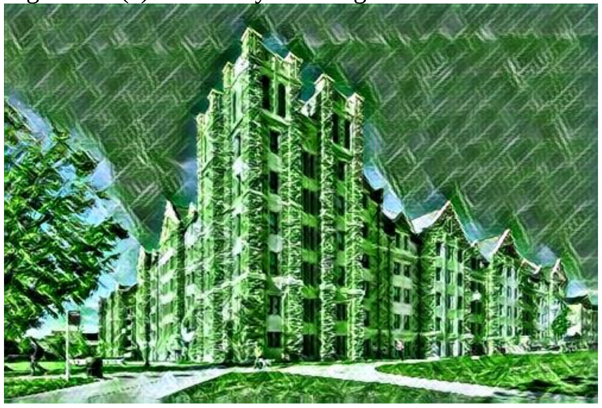


Figure 10. (a) Green Crystal Image


(b)

For the feedforward NST method, it optimizes the required style transferring time in the original NST [5] to real-time, while maintaining the quality of the style transfer results.

4.2.2 Compare Original NST with Feedforward NST

To compare the original NST with the feedforward NST, I tried to make the results become similar by adjusting the learning rate (set to 1e-1) in the original NST and training on VGG16. For the original NST results, see Figures 11, 12, and 13; for the feedforward NST results, see (b) in Figures 8, 9, and 10.

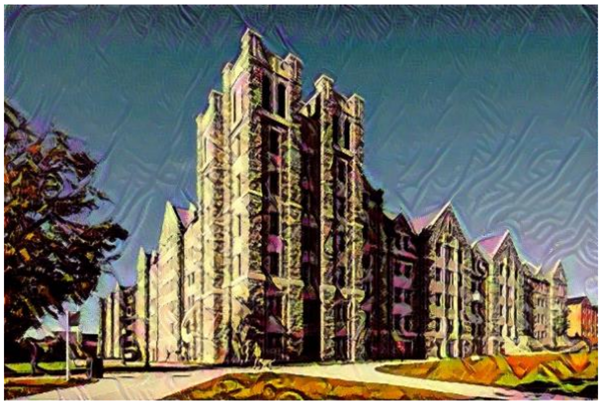


Figure 11.

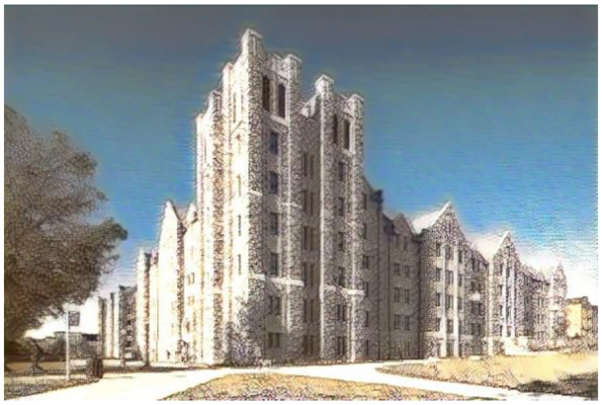


Figure 12.

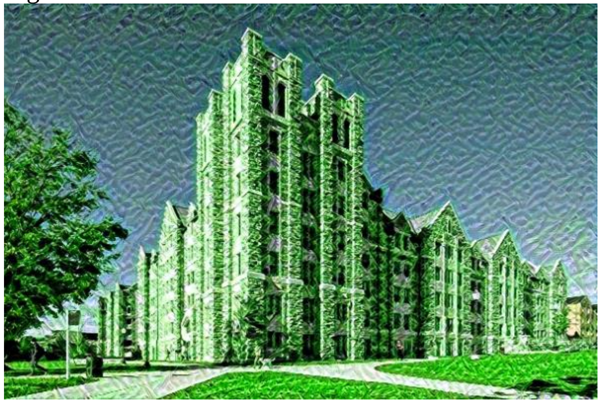


Figure 13.

4.3 CLIP-based method

4.3.1 Number of Cropped Patches

In section **3.2,** we learned the L_{total} which is corresponding to the L_{patch}, moreover, the patch loss is used to preserve the texture of the input content image to achieve more localized style transfer effects.

As the patch loss weight λ_p becomes lower, the influence of the input text condition also gets lower and may result in a stylized image that preserves more of the original texture from the content image. Thus, the user needs to tune the patch loss weight to get the most desired result.

For this experiment, I chose Olive Trees by Van Gogh as the content image and “White Wool” as the text condition. See the results in Figure 14. (b, c, d), we can see (b) has some branch texture blended with the “White Wool” text condition, and (d) shows the richest style from the input text condition.

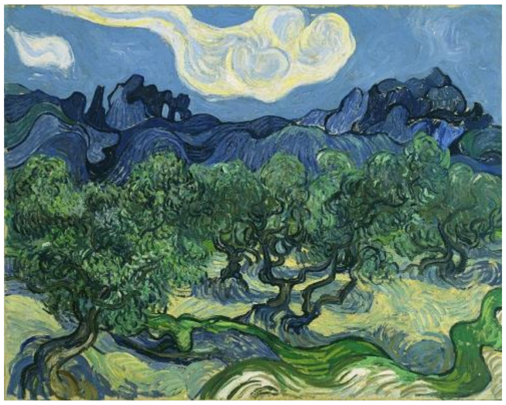
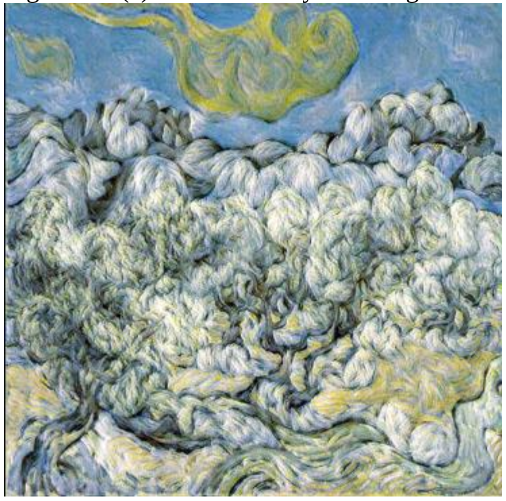
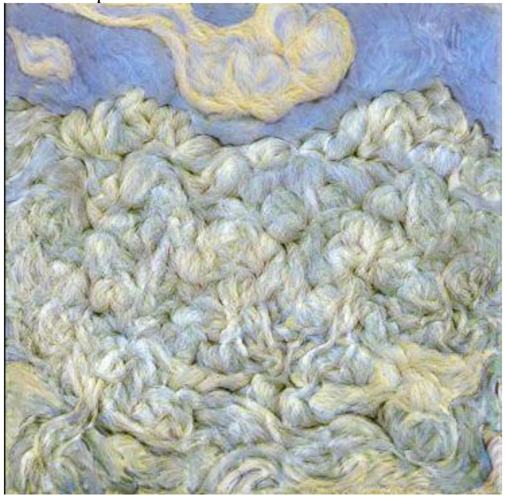
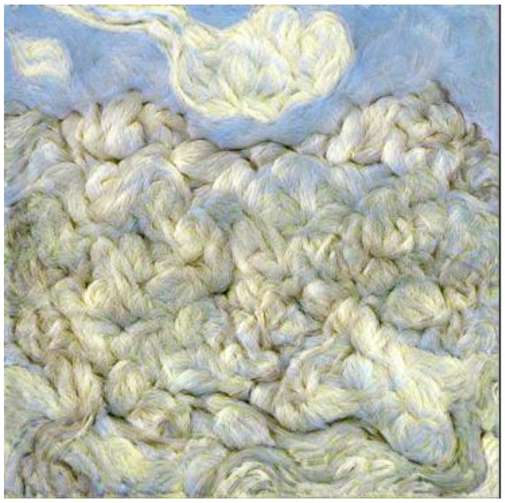


Figure 14. (a) Olive Trees by Van Gogh


(b) Set λ_p to 3000


(c) Set λ_p to 5000


(d) Set λ_p to 9000

4.3.2 Different Text-Content Image Pairs

In the CLIP-based experiments, since there is no need to provide a style image for style transferring, I experimented on different text conditions with several types of content images such as portrait, building, and painting, under the same configuration, see Figure 15.

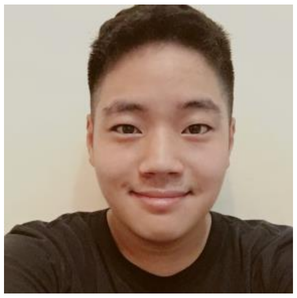
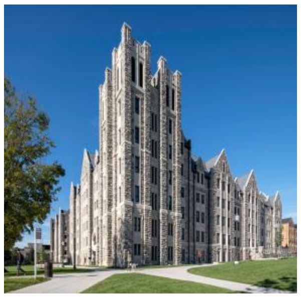
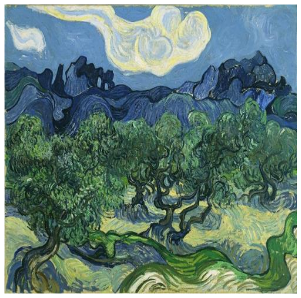


Figure 15. (a) My portrait


(b) Pearson Hall from Virginia Tech


(c) Olive Trees by Van Gogh

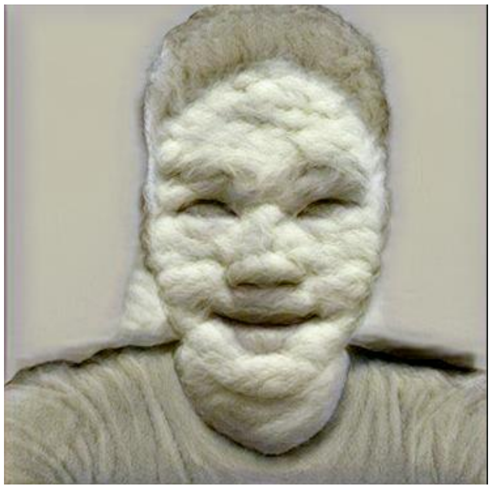


Figure 16. (a) “White Wool”

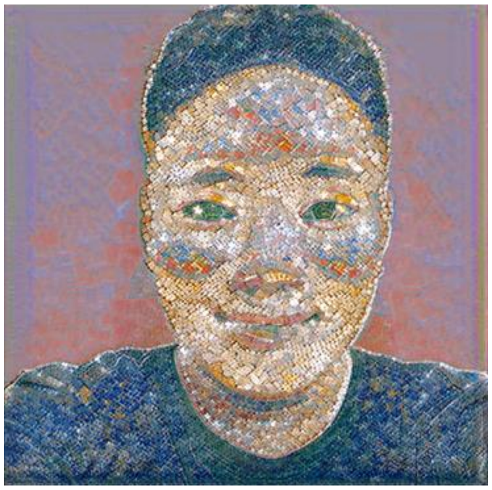
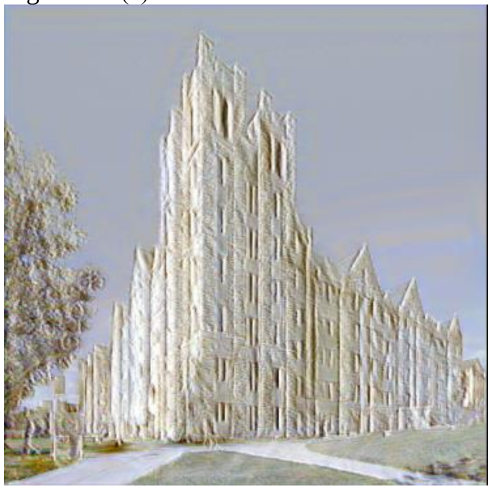
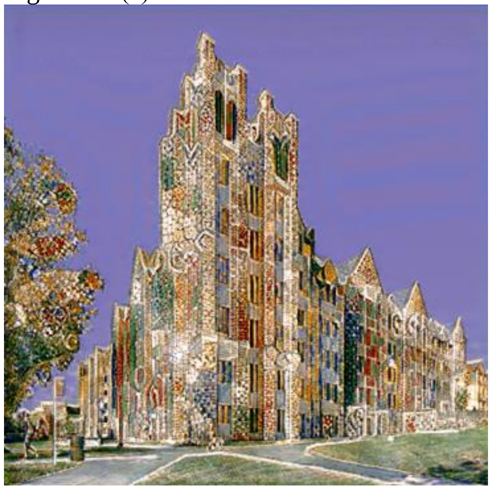
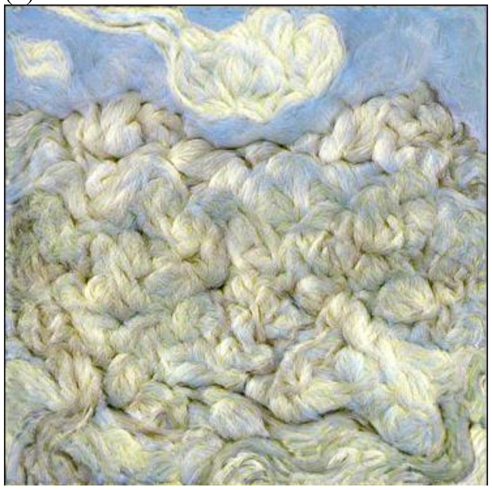
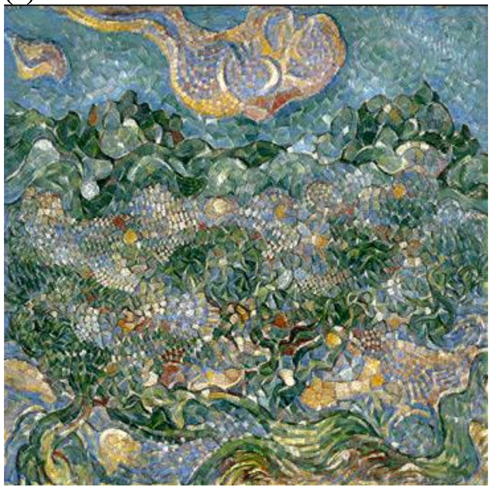


Figure 17. (a) “Mosaic”


(b) “White Wool”


(b) “Mosaic”


(c) “White Wool”


(c) “Mosaic”

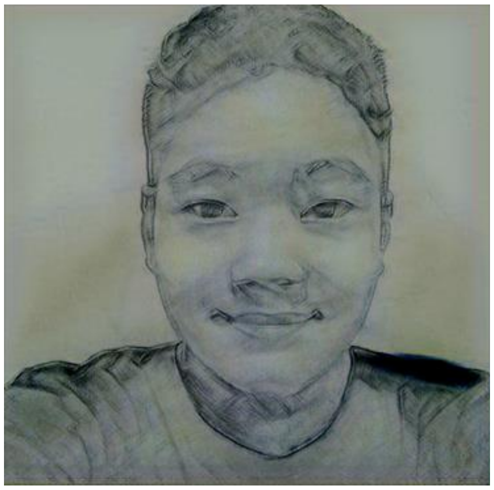
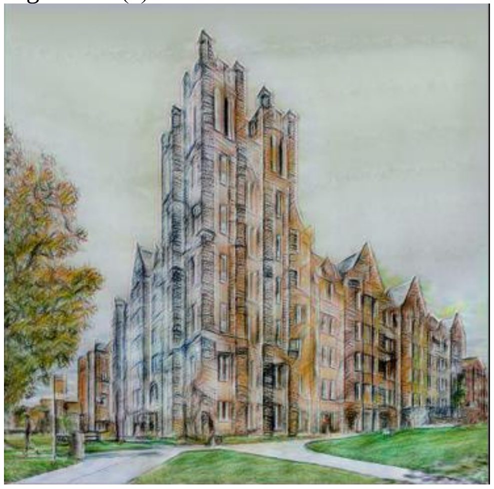


Figure 18. (a) “Sketch with Black Pencil”


(b) “Sketch with Black Pencil”

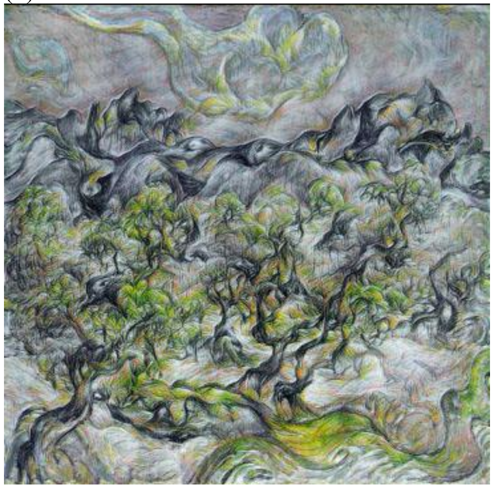

(c) “Sketch with Black Pencil”

Figure 16, 17, 18., show the results with different text conditions, “White Wool”, “Mosaic”, and “Sketch with Black Pencil” respectively, results are corresponded with the content images in Figure 15.

4.3.3 Creative Text Condition

Unlike the original NST and the feedforward NST, the user can customize the input text condition arbitrarily. Thus, we can combine “Watercolor Paint” with “Flower” or “Ink wash painting” with “Green brush” as the input text condition without finding a real artwork. In general, we can specify different styles, objects, and colors to get the desired output. See Figure 19. for the results with Pearson Hall from Virginia Tech as the content image.

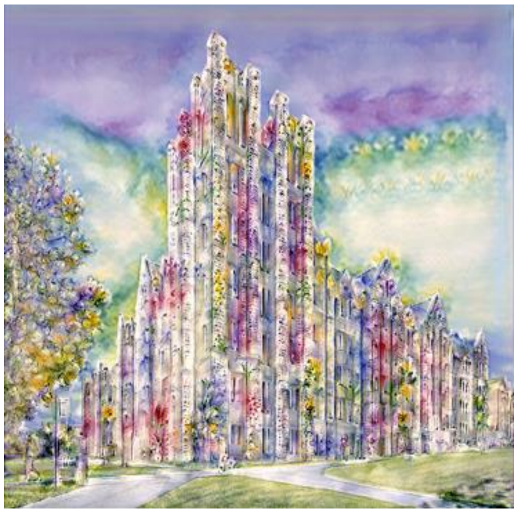
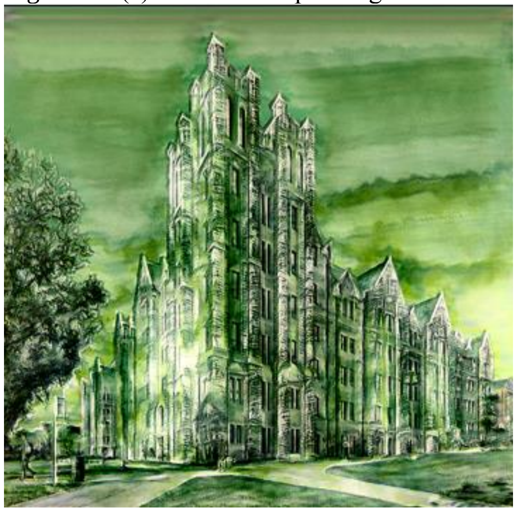


Figure 19. (a) “Watercolor painting with flowers”


(b) “An ink wash painting with a green brush”

4.4 Compare Three Style Transfer Methods

To compare the three methods in this report, I chose the styles that are available to all of them such as green crystal, Starry Night by Van Gogh, and the Great Wave by Hokusai, see Figure 10. (a), 3. (b), and 5. (a) for the style image respectively. Figures 20, 21, and 22 show the results in the following order: (a) displays the original NST image, (b) displays the feedforward NST image, and (c) displays the CLIP-based method image.

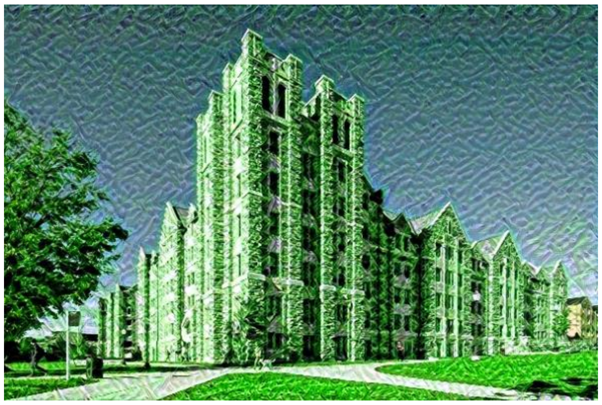
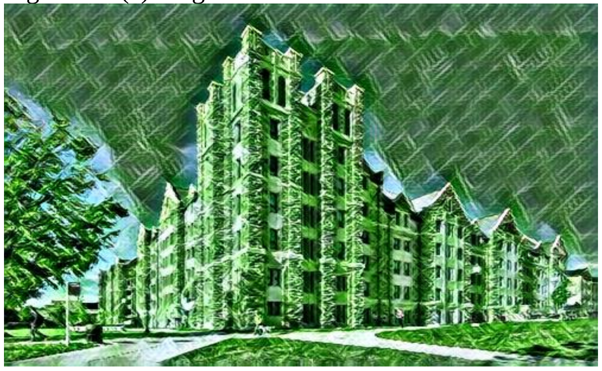
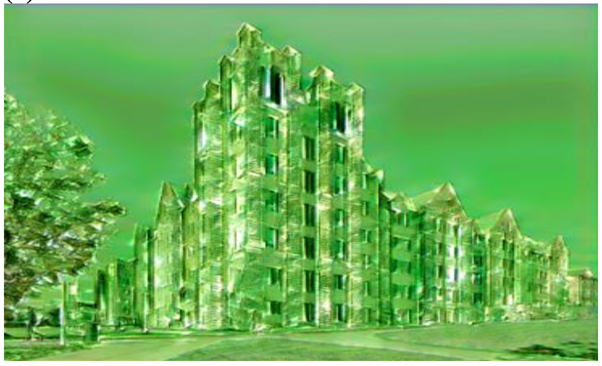


Figure 20. (a) Original NST


(b) Feedforward NST


(c) CLIP-based method with “Green Crystal” text condition

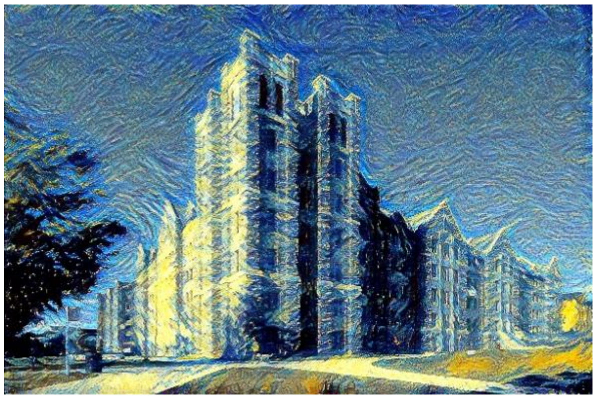
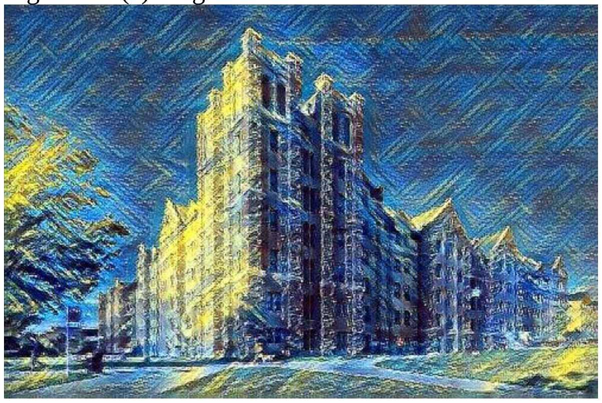
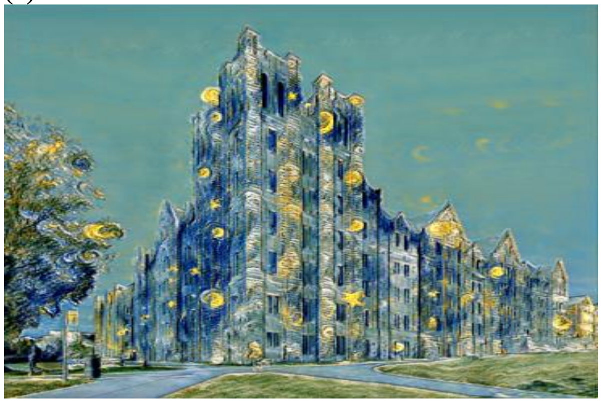


Figure 21. (a) Original NST


(b) Feedforward NST


(c) CLIP-based method with “Starry Night by Van Gogh” text condition

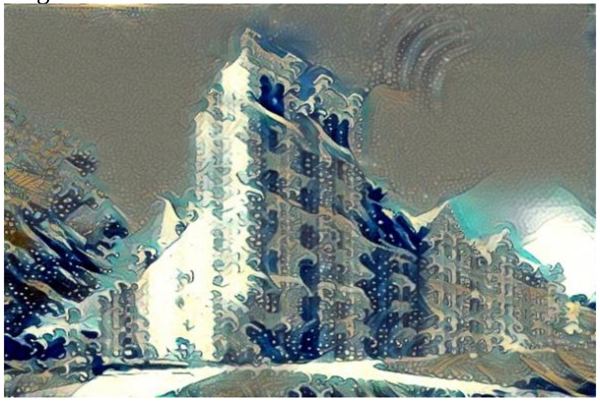
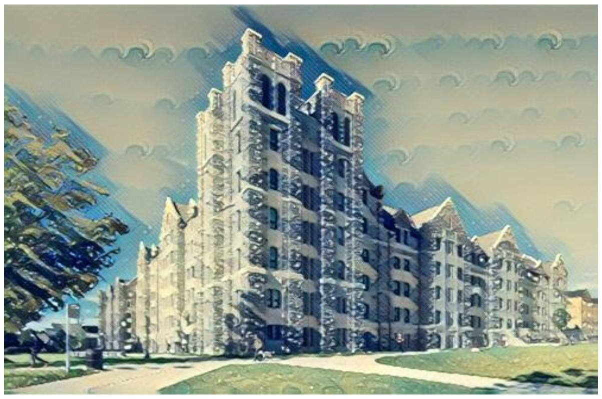
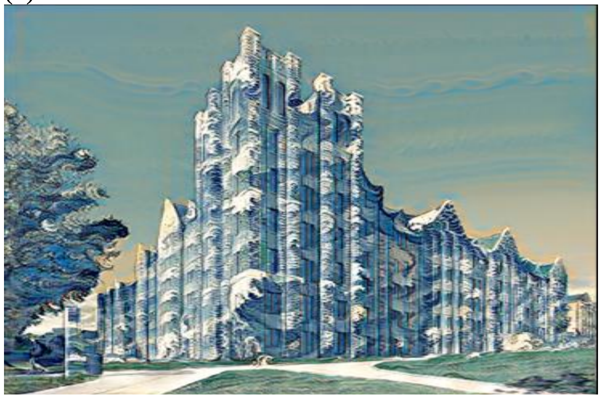


Figure 22. (a) Original NST


(b) Feedforward NST


(c) CLIP-based method with “The great wave of Kanagawa by Hokusai” text condition

4.5 RGB Histogram Analysis

To further discuss the results from an analytical perspective rather than comparing them visually, I analyze the RGB histogram of each style image and the results in the previous section **4.4**. For the RGB histogram of the content image, see Figure 23., style images with results from three methods, see Figures 24, 25, and 26 (ONST, FNST, and CLIP stands for original NST, feedforward NST, and CLIP-based method respectively).

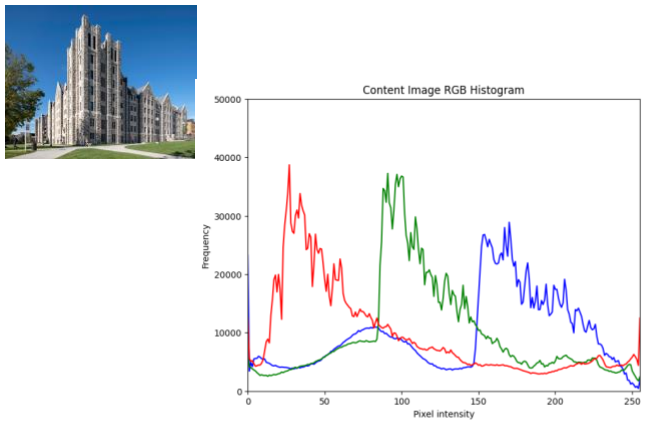


Figure 23. Pearson Hall from Virginia Tech

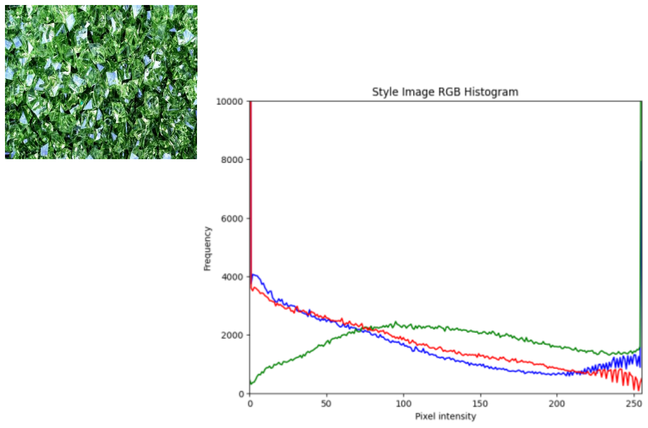
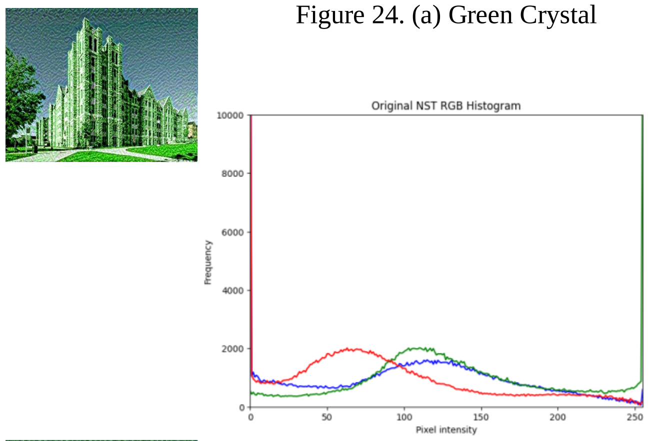
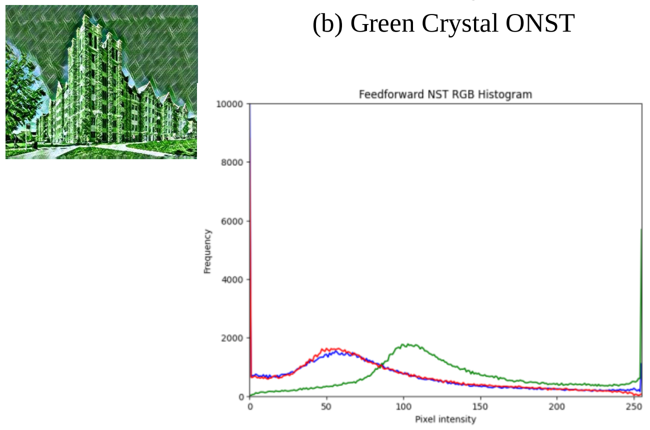


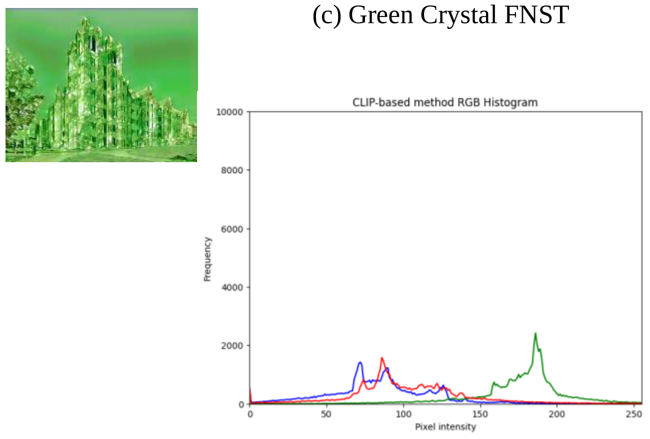
Figure 24. (a) Green Crystal



(b) Green Crystal ONST



(c) Green Crystal FNST



(d) Green Crystal CLIP

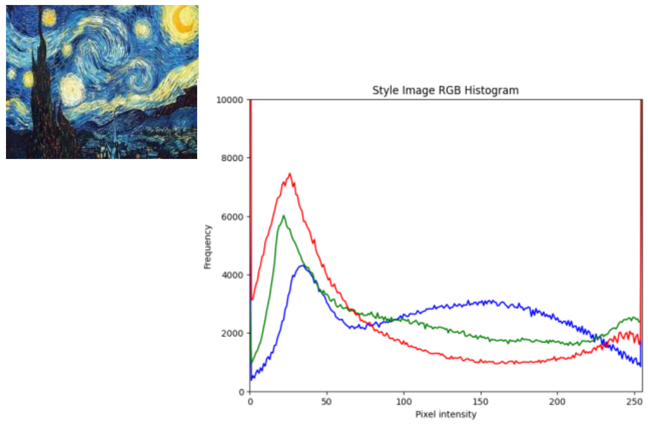
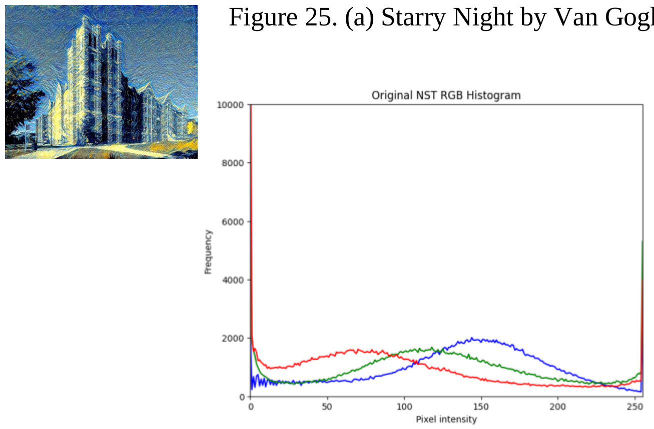
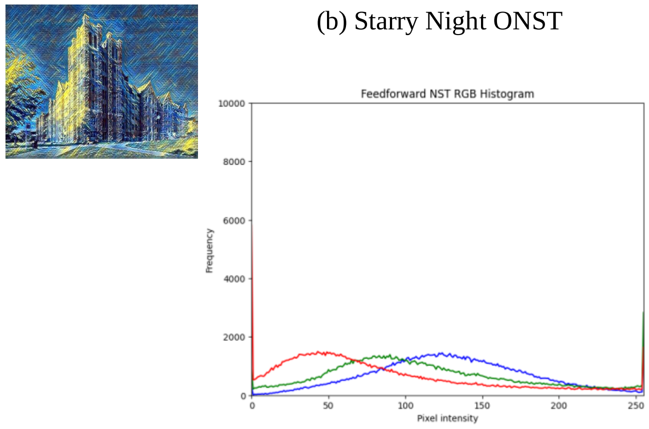


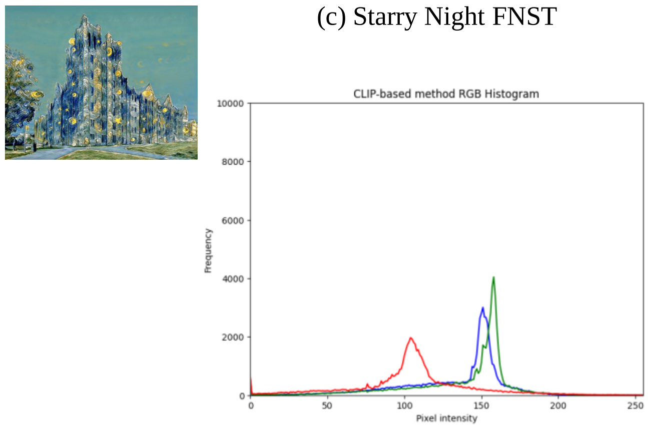
Figure 25. (a) Starry Night by Van Gogh



(b) Starry Night ONST



(c) Starry Night FNST



(d) Starry Night CLIP

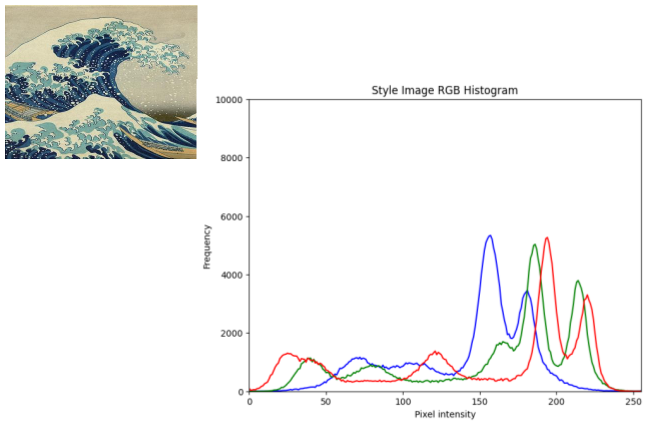
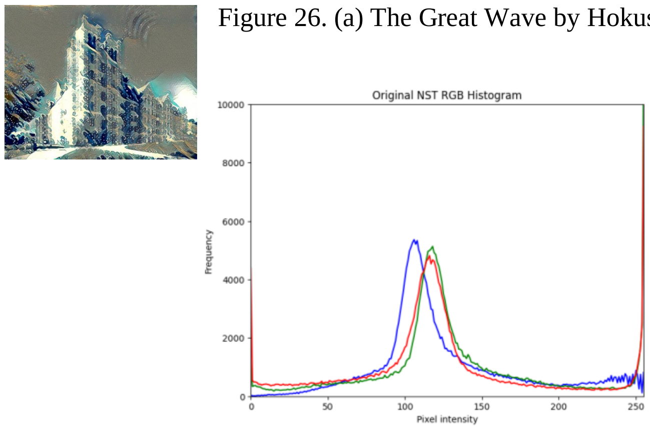
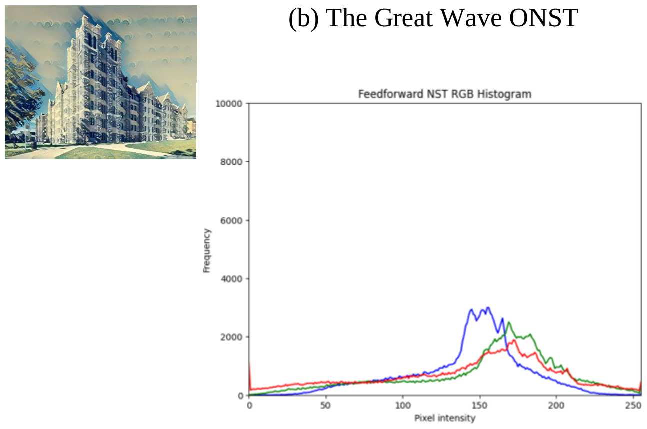


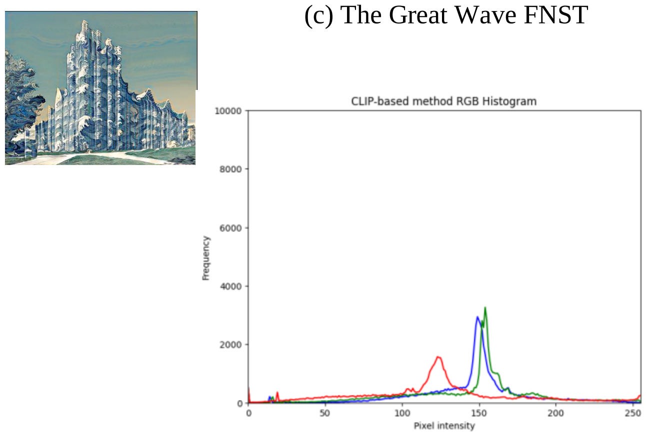
Figure 26. (a) The Great Wave by Hokusai



(b) The Great Wave ONST



(c) The Great Wave FNST



(d) The Great Wave CLIP

From the above images, we can find that the RGB histograms from the original NST and the feedforward NST are like each other, which is as we expected since the ways of extracting the features from the content and style image are similar in both methods. However, the results from the feedforward method not only have a higher overall RGB histogram similarity but also indicate the feedforward method can provide more stable results once we pre-trained the style transfer network. Nevertheless, the original NST method is more agile since the users can change the weight and parameters of the coming optimized process.

For the CLIP-based method, it is hard to give a thorough insight since we do not know what exactly style images were learned by the CLIP model before; however, we can still find that the RGB histograms look like each other even with three completely different text conditions. Moreover, the results barely have a frequency at pixel intensity 255 and 0, which usually means white and black pixels, which show the resulting images have been blended evenly and stable over different style targets.

4.6 Double Style Transfer

In this experiment, I took The Scream by Edvard Munch and the Candy Style Image [15] as the style images, see Figure 27. (a, b), and Pearson Hall from Virginia Tech as the content image. Also, by adjusting the ratio for the two style images, we can easily observe the difference in the result. For the single style transfer results, see (c, d) in Figure 23; for the double style transfer, see (e, f, g) in Figure 23.

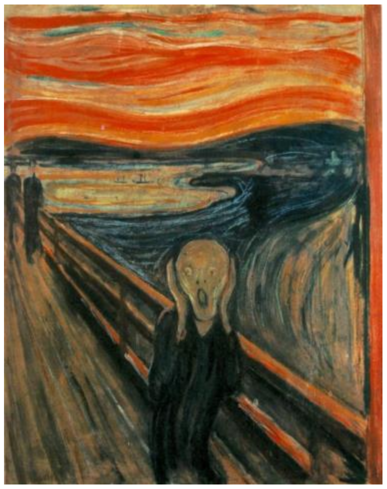
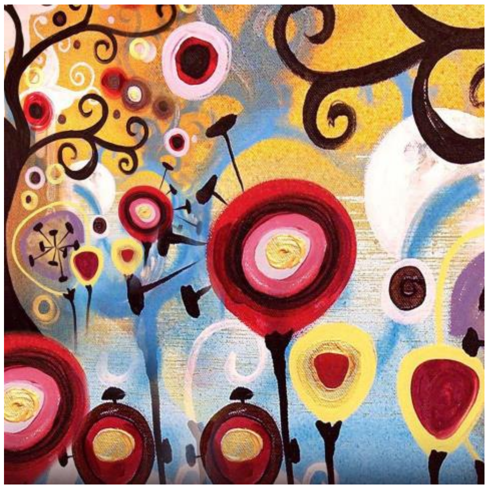
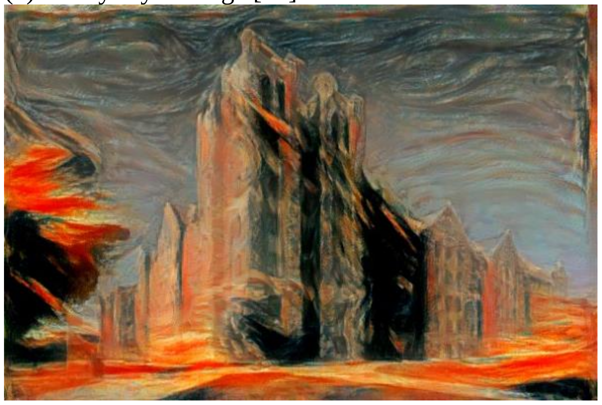
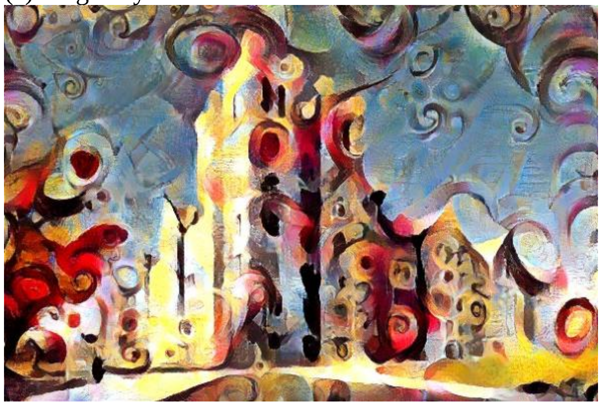


Figure 27. (a) The Scream by Edvard Munch


(b) Candy Style Image [15]


(c) Single style transfer result on The Scream


(d) Singe style transfer result on the Candy

(e) Ratio between two styles, The Scream:Candy=3:7

(f) Ratio between two styles, The Scream:Candy=5:5

(g) Ratio between two styles, The Scream:Candy=7:3

5. Conclusion

In this project, I did style transfer tasks with three different methods, the original NST [5], the feedforward [18], and the CLIP-based method [20]. With the previous experiments and the understanding of the history of style transfer in the computer vision realm, here are my personal guidelines for the readers and future studies to lookup:
(1) Do you have style images or not? If you do not have a style image, then use the CLIP-based method. Otherwise,
(2) First of all, use the original NST method and play with different parameters such as content weight and style weight, total variation loss, layers to extract features, and optimizer.
(3) Once you get desired results, you can train a network for the style image with the knowledge you have from the previous step (2). Also, you can further tune the model by adjusting the learning rate.
(4) Try on the CLIP-based model with any text conditions and experiment with different numbers of patches, directional CLIP loss, total variation loss, and content loss.

Finally, I hope this project has shed some light on the versatility of style transfer and its potential to generate creative and pleasing content.

References

[1] A. Efros and T. K. Leung. Texture synthesis by nonparametric sampling. In Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on, volume 2, pages 1033–1038. IEEE, 1999.

[2] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pages 341–346. ACM, 2001.

[3] Matthew D Zeiler, Rob Fergus. "Visualizing and Understanding Convolutional Networks," in arXiv 2013.

[4] Leon A. Gatys, Alexander S. Ecker, Matthias Bethge. "A Neural Algorithm of Artistic Style," in arXiv 2015.

[5] L A. Gatys, A. S. Ecker and M. Bethge, "Image Style Transfer Using Convolutional Neural Networks," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2414-2423, doi: 10.1109/CVPR.2016.265.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'12). Curran Associates Inc., Red Hook, NY, USA, 1097–1105.

[7] Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. 3rd International Conference on Learning Representations (ICLR 2015), 1–14.

[8] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770-778.

[9] J. Hu, L. Shen and G. Sun, "Squeeze-and-Excitation Networks," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 7132-7141, doi: 10.1109/CVPR.2018.00745.

[10] Egan, B. (2018). neural-style-pt (Version 1.0.0) [Computer software]. https://github.com/ProGamerGov/neural-style-pt

[11] LIU, D. C. & NOCEDAL, J. (1989). On the limited memory BFGS method for large scale optimization. Math. Programming, 45, 503--528.

[12] Kingma, D. and Ba, J. (2015) Adam: A Method for Stochastic Optimization. Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015).

[13] Ragab DA, Sharkas M, Marshall S, Ren J. Breast cancer detection using deep convolutional neural networks and support vector machines. PeerJ. 2019 Jan 28;7:e6201. doi: 10.7717/peerj.6201. PMID: 30713814; PMCID: PMC6354665.

[14] https://towardsdatascience.com/mixed-neural-style-transfer-with-two-style-images-9469b2681b54

[15] https://github.com/pytorch/examples/blob/main/fast_neural_style/images/style-images/candy.jpg

[16] H. Lee, S. Seo, S. Ryoo, and K. Yoon. Directional Texture Transfer. In Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering, NPAR '10, pages 43–48, New York, NY, USA, 2010. ACM.

[17] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image Style Transfer Using Convolutional Neural Networks," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, Jun. 2016, pp. 2414–2423. doi: 10.1109/CVPR.2016.265.

[18] Perceptual Losses for Real-Time Style Transfer and Super-Resolution by Justin Johnson, Alexandre Alahi, Li Fei-Fei, 2016

[19] Learning Transferable Visual Models From Natural Language Supervision by Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, Ilya Sutskever, 2021

[20] CLIPstyler: Image Style Transfer with a Single Text Condition by Gihyun Kwon Jong Chul Ye, 2022

[21] Gordić, Aleksa, pytorch-neural-style-transfer, 2020, https://github.com/gordicaleksa/pytorch-neural-style-transfer

[22] Gordić, Aleksa, pytorch-neural-style-transfer, 2020, https://github.com/gordicaleksa/pytorch-nst-feedforward

[23] Dong, C., Loy, C.C., He, K., Tang, X.: Image super-resolution using deep convolutional networks. (2015)

[24] Cheng, Z., Yang, Q., Sheng, B.: Deep colorization. In: Proceedings of the IEEE International Conference on Computer Vision. (2015) 415–423

[25] Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. CVPR (2015)

[26] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. arXiv preprint arXiv:2103.00020, 2021

[27] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylegan imagery. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 2085–2094, 2021.

[28] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In IEEE Conf. Comput. Vis. Pattern Recog., pages 4401–4410, 2019

[29] Rinon Gal, Or Patashnik, Haggai Maron, Gal Chechik, and Daniel Cohen-Or. Stylegan-nada: Clip-guided domain adaptation of image generators. arXiv preprint arXiv:2108.00946, 2021

[30] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylegan imagery. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 2085–2094, 2021.

[31] Rudin, Leonid I. et al. "Nonlinear total variation based noise removal algorithms." *Physica D: Nonlinear Phenomena* 60 (1992): 259-268.