# Neural Style Transfer Using VGG19

Nathan Warren

Duke
PRATT SCHOOL *of* ENGINEERING

## Introduction

Convolutional neural networks (CNNs) are commonly used for object detection where they can learn both lower level and higher level features from images. Higher level features, such as shape and global arrangement of an image can be defined as content while the lower-level features, such as pixel colors and edges, can be defined as the style of an image. Explored by Leon Gatys *et al.*, in his 2016 paper, "Image Style Transfer Using Convolutional Neural Networks," Gatys *et al.,* showed that representations of content and style are indeed separable in the convolutional neural network. Since content and style are separable, they can be combined to form hybrid images.

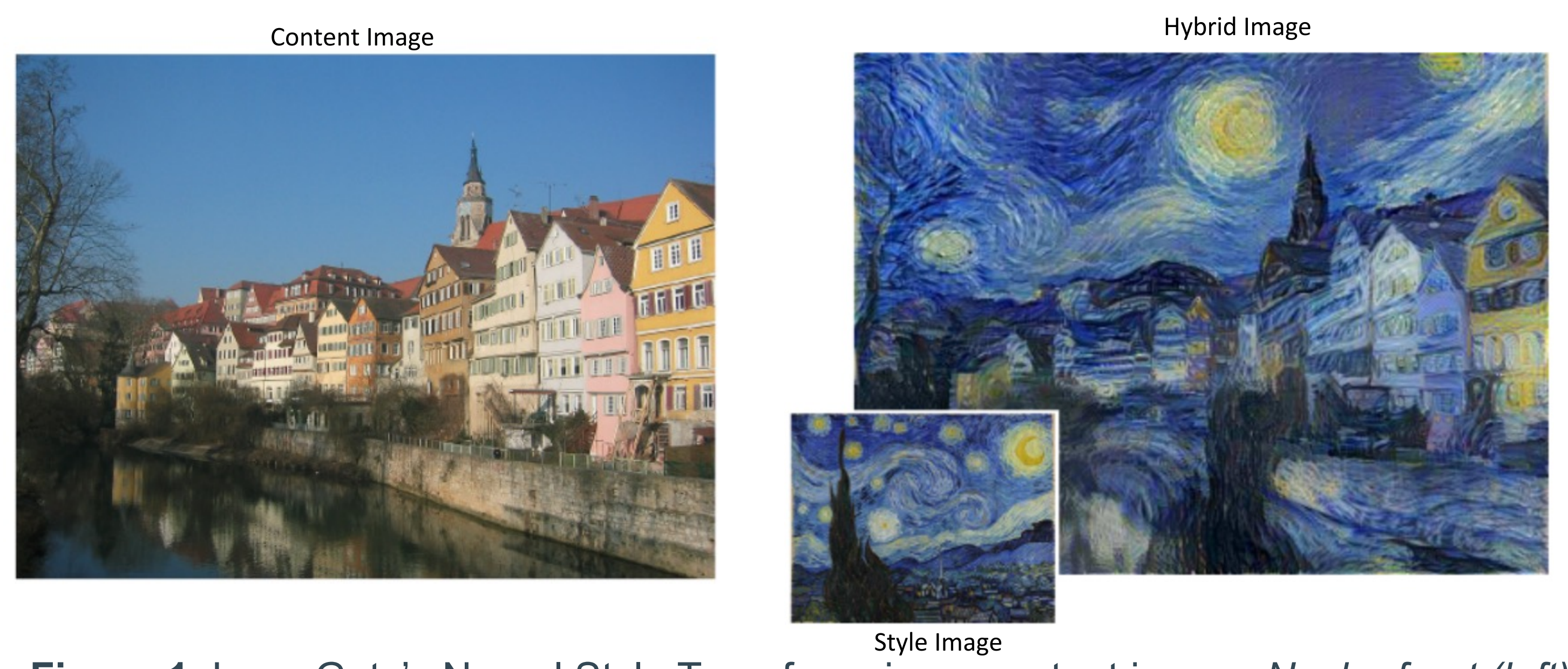

**Figure 1**: Leon Gaty's Neural Style Transfer using a content image, *Neckarfront (left),* and a style image of Van Gogh's *Starry Night (bottom)*.

## Goals

- Replicate Gaty *et al.* Neural Style Transfer work
- Test Images of different complexity for both style and content
- Explore how changing style layers and weighting will affect the output hybrid image

## Methods

Pre-trained VGG19 was used and all max pooling layers were replaced with average pooling layers. In order to get features from an image, that image had to be run through the pre-trained VGG19 network. Forward hooks were used to extract feature maps from certain layers. A special numbering system is used to identify convolutional layers based on pooling layers. 1-1 would indicate convolutional layer 1 while 2-2 would indicate convolutional layer 2 directly after the first pooling operation.

**Content**

Content of an image is extracted from deeper layers as they contain features more related to shapes and the spatial arrangement of the image. Layer 4-2 was used here.

**Content Loss**

By running our content image through the network and getting the feature map at conv 2-2, we can take the square error loss between the content layers for the original and generated image.

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

**Style**

Style of an image is extracted from earlier layers as they contain more information regarding colors and edges of an image. The style layers taken were 1-1, 2-1, 3-1, 4-1, and 5-1. A gram matrix is used to find the correlation between the features in all of these layers.

**Style Loss**

Taking the inner product generated by the gram matrix, we can define our loss function for style as shown below where G is the generate style image at layer l, channel j, and A is the original style image.

$$E_l=\frac{1}{4N_l^2M_l^2}\sum_{i,j}\left(G^l_{ij}-A^l_{ij}\right)^2$$

Weighting can be applied to each layer to give us the following style loss.

$$\mathcal{L}_{\text{style}}(\vec{a},\vec{x})=\sum_{l=0}^{L}w_lE_l,$$

**Creating Hybrid Images**

Creating hybrid images is an optimization problem between the two images. In order to create this optimization problem, since a loss for content and style has been create, we can create a total loss value and seek to optimize it to produce a hybrid image. Alpha and beta are used as hyperparameter to manipulate the representation of style and content in the hybrid image.

$$\mathcal{L}_{total}=\alpha\mathcal{L}_{content}+\beta\mathcal{L}_{style}$$

As the original content image is iteratively run through the network, the gradient is used to update the pixel values to minimize the loss.

## Results

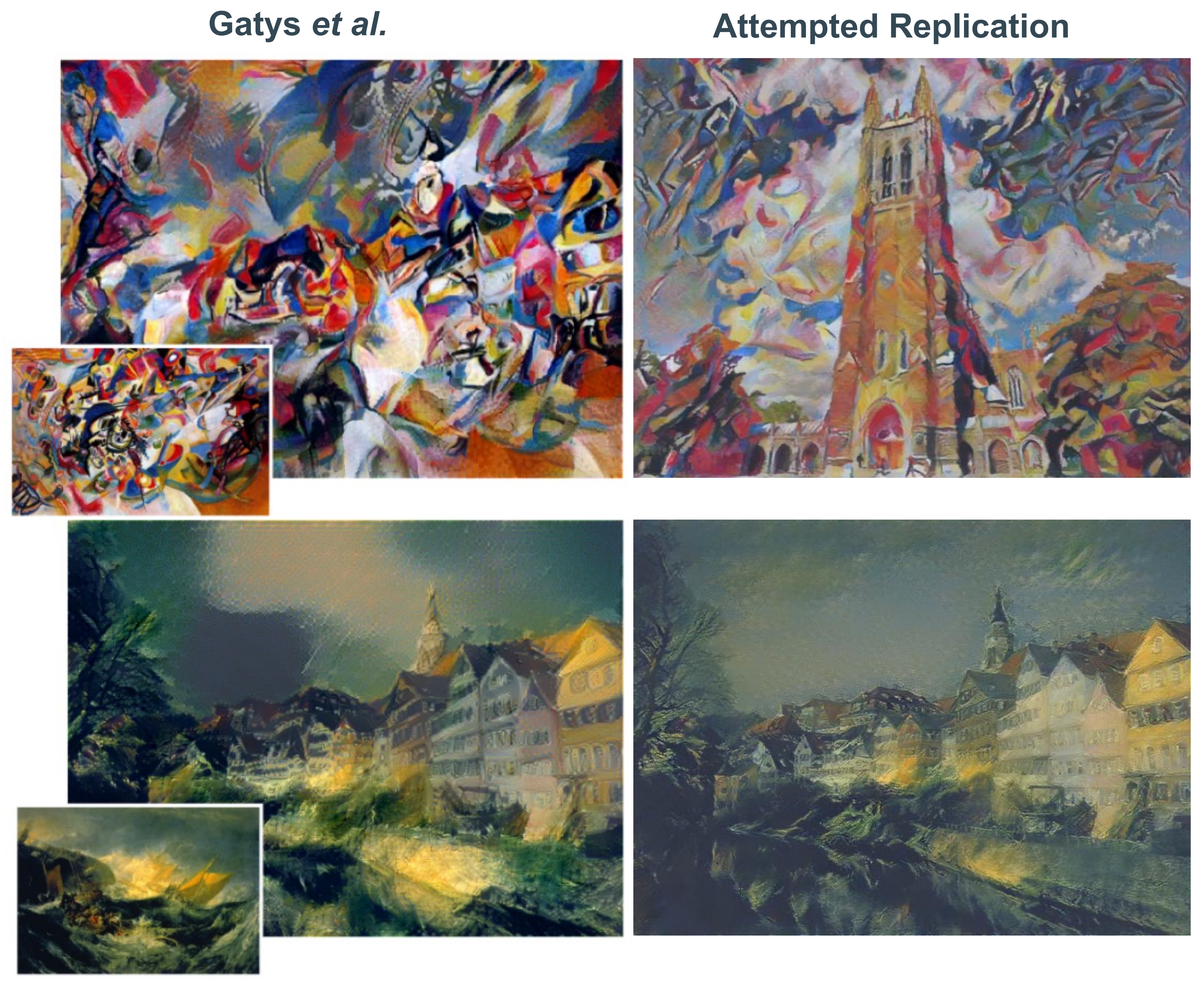

**Figure 2**: Gaty's original hybrid images( left) with style image (bottom left), attempted replication by me (right). While style has been transferred, there appears to be differences in contrast. Images were run for 30 epochs using the optimizer LBGFS. α = 1, β = 1e4.

## Different Image Complexities

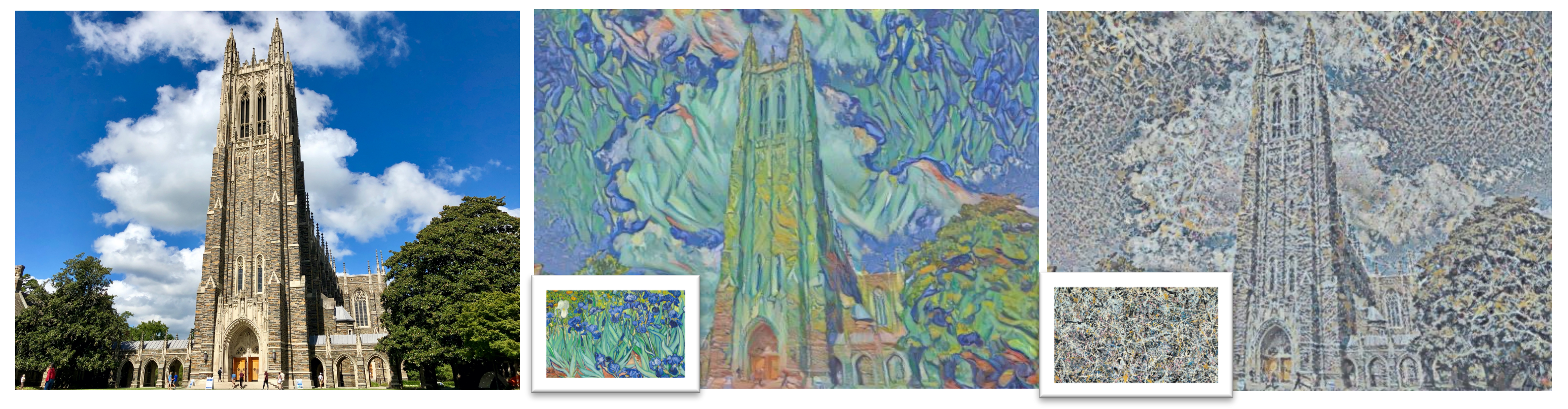

**Figure 3**: Original image of Duke Chapel (left). Chapel image styled with Van Gogh's, *Irises* (middle), Chapel image styled with Jackson Pollock's *Number 1* painting. All style transfers worked relatively well despite the change in complexity of the style image, although the less complex style image worked better overall.

## Different Style Layers Used

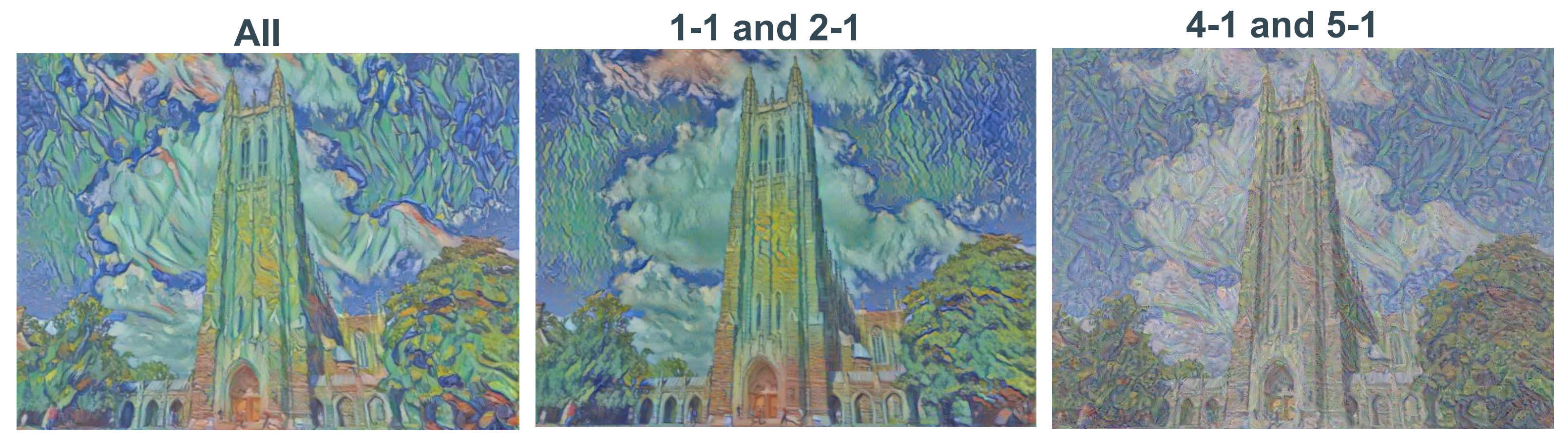

**Figure 4:** The chapel image and Van Gogh's, *Irises* were used to create all the images above. The first image contains all style layers. The middle image has been styled with only layers 1-1 and 2-1 (equally weighted), from the style image. The image on the right has been styled with only layers 4-1 and 5-1 (equally weighted) from the style image.

## Conclusions

My implementation of NST works although there may be color normalization issues based on the contrast seen on hybrid images compared to style images. Complexity of the style photo can sometimes play a role in how hard a style is to transfer as seen in Figure 3. The less complex photo had its style transferred in a way that was much more aesthetically pleasing although style transfer did still occur with the more complex style image. Even though my work is not an exact replicate of Gatys *et al.* work, style was still transferred. It also appears that style images with "texture" transfer more efficiently which may explain why Gatys *et al.* explained that transferring style from photos did not work well.

Based on my experiment into applying different style layers to content images, it appears that color is extracted earlier on and edge like shapes are extracted later in VGG19. Sharpness of the style appears to come from later layers. Future work includes improving the style transfer and attempting to make style transfer real-time for video use.

### References

1. Gatys, L., Ecker, A., & Bethge, M. (2016). A Neural Algorithm of Artistic Style. *Journal Of Vision*, *16*(12), 326. doi: 10.1167/16.12.326
2. PytorchNeuralStyleTransfer. (2020). Retrieved 21 November 2020, from https://github.com/leongatys/PytorchNeuralStyleTransfer
3. Museum of Contemporary Art. (2020). *Number 1* [Image]. Retrieved from https://www.moca.org/collection/work/number-1
4. Neural Transfer Using PyTorch — PyTorch Tutorials 1.7.0 documentation. (2020). Retrieved 21 November 2020, from https://pytorch.org/tutorials/advanced/neural_style_tutorial.html