Exploring Neural Style Transfer

Ryan Qiu(rtq), Erik Hoberg (ehoberg), Richard Jiang (rjiang98)

Introduction

In 2015 Gatys et al. created a method of transferring artistic style from one image to another they called Neural Transfer. As appreciators of art, we were amazed by their results and wanted to explore improvements in art generation. By working on this project, we furthered our understanding of Neural Style Transfer as well.

We think this is really interesting and potentially a new movement of art. Our work here is aimed at expanding the creativeness of style transfer.

Neural Style Transfer

Gatys et. al. takes specific convolutional layers of VGG-19 and uses these layers in creating a new image that combines a content image with a style image. They complete this by running an image through VGG-19 and then evaluating how well that image's content and style matches the original content and style images respectively. They quantify this by defining a loss function composed of a content score and a style score:

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

where $\vec{p}$ is the content image, $\vec{a}$ is the style image, and $\vec{x}$ is the output image.

The content loss is defined to be the mean-squared error of the feature mappings of the input image and the output image

The style loss is the mean-squared error between the Gram Matrix of output image and the style image at specific layers of VGG.

Using this loss function, we update the pixels of our image and continuously run gradient descent for our output image to best match the style and content of the input images.

References

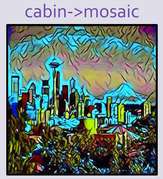
[1] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A Neural Algorithm of Artistic Style. 2015. URL: http://arxiv.org/abs/1508.06576.

Multi-Style Transfer

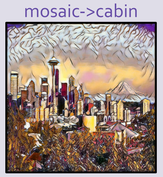
Sequential Style Transfer

One possible way to incorporate multiple styles is to run style transfer from one image onto the content image, then use that resulting image as the content image for another pass with a different style. What we see is that the final result is heavily weighted towards the second style image.

cabin->mosaic



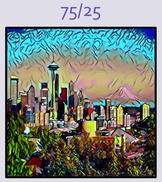
mosaic->cabin



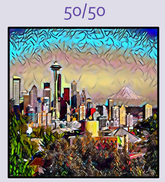
Weighted Style Transfer

An improved approach was to use both style images simultaneously in our model. By combining both style scores into our loss function as we train, our output image mixes both. We can also weight the contribution of the style score of each style image, we can shift the image to appear more like one style than the other.

75/25



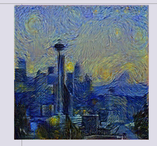
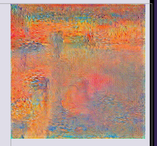
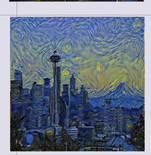
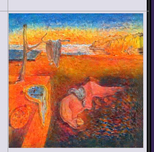
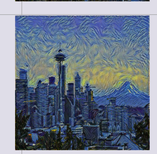
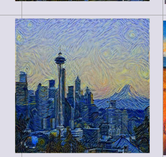
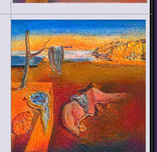
50/50



25/75

Different Architecture

One problem we identified is that the neural style transfer algorithm often takes a significant amount of time to produce images of reasonable size. We explored different network architectures to identify if we could reduce the time this algorithm took to run without significantly affecting the quality of the output image. Each image output was run for 600 iterations on a GPU with a constant style vs. content weight ratio. The content layer and style lsyers are chosen in the same manner as in Gatys et. al.

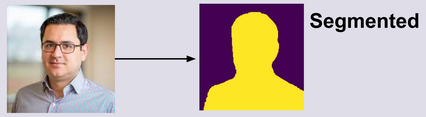
Network	Time(s)		
AlexNet	17.36		
VGG-11	32.99		
VGG-16	46.61		
VGG-19	51.57		

The results show a fairly promising trend that the time taken by this algorithm can be reduced while still producing reasonably aesthetic images. However, it seems that something like AlexNet may not be complex enough for this task, as detrimental artifacts are present (ie.criss-crossing patterns or blurred output)

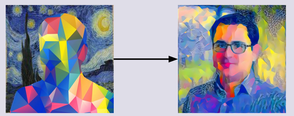
Localized Style Transfer

An extension to multi-style transfer is to control where the style is transferred to. We accomplished this in two ways.

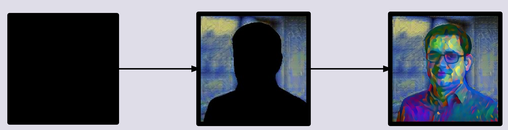
The first step for both methods was to perform semantic segmentation the content image to get a 1D "mask" tensor where each coordinate represents the class that a particular pixel belongs to. We used a pretrained DeepLabv3 model to perform the segmentation.



The first method was to use our segmented image to stitch together a new style image. Each unique class' position in the segmentation mask will be filled with one particular style. We then use this stitched style image as to stylize our content image. We can see below that this does not give us a clean divide between styles, but it is fast and styles are partially localized according to the segmentation mask.



The second method was to stylize our content image once per class that appears in our segmentation mask. Then, whenever a particular class appears in the segmented image, we fill that pixel in with the RGB value of one of the styled images. You can see below that this works well, although it is very slow as we are styling multiple images



Future Work

1. Exploring localized style transfer on segmented classes. For example, we could style the trees of one image with the style of the trees in a reference image.
2. Instead of using VGG, we could create a more niche art detection network. The layers of this network might better identify the defining characteristics of various art styles, and result in a better style loss