Deep Photo Style Transfer

William Hogan[1], Cole Feng[1], Ramkesh Renganathan[2], Vincent Nguyen[1], Luke Rohrer[1], and Varun Viswanath[1]

[1]Department of Computer Science and Engineering, University of California San Diego, San Diego, CA 92093
[2]Viasat, Carlsbad, CA 92009

Abstract

In this project, we use PyTorch to reproduce the experiments described in *Deep Photo Style Transfer* [9]. Photographic style transfer seeks to transfer the style of one image onto another. Through the use of a pre-trained image classifying neural network, it is possible to extract style features of one image and apply them to the content, or scenery, of another image. The approach builds on a previous style-transfer model suggested in *A Neural Algorithm of Artistic Style*[3] by increasing the photo-realism of the generated images. We expanded the scope of the project by comparing the model from *Deep Photo Style Transfer* [9] with two high performing generative adversarial networks, namely CycleGAN and Multimodal Unsupervised Image-to-Image Translation(MUNIT).

We were able to generate high-quality images using the style-transfer model. We found that image segmentation and the matting laplacian greatly increased the photo-realism of our results. The strength of the style-transfer model is that it is able to merge any two photos together but the best results came from images that shared similar features such as a painting of a landscape paired with a photo of a landscape. The CycleGAN and MUNIT models were slightly more restrictive in that they required images to share the same features and be from the same domain (e.g. photo to photo instead of painting to photo). The CycleGAN and MUNIT models performed well transferring styles between photos, but they failed to capture both style and content when one domain was art and the other was a photo.

1 Introduction

The unsupervised learning task of Image Style Transfer aims at changing the style of an image while preserving its content. As suggested in [4], this task usually has two contexts: One is example-guided style transfer, in which the target style comes from a single sample, and collection style transfer, in which the target style is defined by a collection of images.

The focus of our final project will be on the example-guided style transfer task. We will recreate the experiment described in *Deep Photo Style Transfer* [9]. For example, in Figure 1 the reference style of the middle image is applied to the left image to produce the right image, which displays the same scene as the input but with the style of the reference photo. By altering the reference image, one can drastically change a photo to to make it appear as if it were taken under different circumstances (illumination, time of day, season, etc.) or portray it in a different artistic style.

We will also explore the collection style transfer task. More specifically, we will re-implement the code of CycleGAN [12] and *Multimodal Unsupervised Image-to-Image Translation* [4] to perceptually compare the results with the former task result.

Figure 1: From left to right, (i) the original photo, (ii) the style-reference photo, and (iii) the generated image which is the scene from the original photo combined with the style of the style-reference photo. [9]

2 Motivation

A style-transfer model has many interesting applications. It can be used to create interesting new art. Casual users can use it to modify personal photos with cool effects or simulate different lighting in their home. Professionals, such as interior designers, makeup artists, and painters can use a style transfer model to visualize their designs. This style-transfer approach is more flexible than a typical GAN. Typical GANs require numerous images from the same domain to properly train a generator and discriminator but this model only requires two images—a style image and a content image. Lastly, the style-transfer model is interesting for educational purposes. Building and experimenting with the style-transfer model helps us learn about and visualize the features extracted by a convolutional neural net. We can leverage that understanding to create new and interesting images.

3 Related Works

This project is a PyTorch recreation of the model and experiments introduced in *Deep Photo Style Transfer* [9]. Luan et al. suggests a deep-learning approach to photographic style transfer that handles a large variety of image content while faithfully transferring the reference style. The approach builds on a previous style-transfer model suggested in *A Neural Algorithm of Artistic Style*[3] by increasing the photo-realism of the generated images.

The image segmentation used in this project was heavily influenced by the work contained in *Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs* [1]. We used image segmentation to match the style of objects in a style image to the corresponding objects in a content image. For instance, with image segmentation, the style of a tree in a style image is *only* applied the trees in a content image. This helped us achiever a higher degree of photo-realism.

A similar style-transfer model is presented in the paper *Universal Style Transfer via Feature Transforms*[7]. They describe a "universal" style-transfer model that it capable of generalizing to unseen styles without compromising visual quality. They present a method that uses a pair of feature transforms, whitening and coloring, that are embedded to an image reconstruction network. The whitening and coloring transforms reflect a direct matching of the given features of the style and content images [7]. They use the same pre-trained VGG19 CNN as we use in this project and, with it, they were able to achieve fairly good results. Since their "universal" model is a style-transfer model mixing two images, they are unable to quantitatively compare their results with other models. Instead, the use a qualitative comparison, similar to the comparison we and Luan at al. do to claim good results. Notably, their model is able to generalize, blending previously unseen style and content images, which is something our model is incapable of.

4 Dataset

For example-guided style transfer, we will focus on recreating the results of *Deep Photo Style Transfer* by using a combination of images from various sources. The combinations of input photos and style-reference photos are endless. We will assemble our own training set of images, drawing from different repositories of images available in the public domain. Our collection of images is assembled from: (a) photos of diverse artwork available from the Museum of Modern Art and the Metropolitan

Museum of Art, including works by Vincent van Gogh and Frida Kahlo, which will primarily serve as reference styles, and images of various idols, sculptures, and figurines, which will serve as inputs, (b) photos of notable landscapes, (c) photos of interior spaces from design catalogs, (d) several personal photos selected for their patterns, textures, or for personal interest, and (e) photos used in the original experiment we are reproducing.

For collection style transfer, we use two datasets, each of which contains two domains of images: **Landscape↔Van Gogh**. We first experiment on the vangogh2photo dataset used by CycleGAN [12]. **Celeb A↔Van Gogh**. We then experiment on a more difficult task where domain A is human faces [8] and domain B is landscape arts by Van Gogh.

5 Methods

Since we are attempting to replicate the results of [6], our input data is the same. For a single transfer, we take a **content** image and a **style** image. Additionally, segmentation maps may be included to constrain the style transfer to entities of the same subject within both images.

There were several other designs that we implemented to compare their efficacy against that of the base model. One of these was the CycleGAN, an approach where content and style images are not directly paired. The generator network is aware of the domain that an image belongs to, but it is allowed to generate different combinations of style and content images.

5.1 Baseline Loss

Here, we give a brief overview of the baseline objective function that we modify from [6]. There are slight differences from the original paper which we find are also present in the original paper's Lua implementation.

$$\mathcal{L}_{total} = \sum_{l=1}^{L_c} \alpha \mathcal{L}_c^l + \sum_{l=1}^{L_s} \beta \mathcal{L}_s^l \tag{1}$$

In the previous equation, α is used to weight **content loss**, which we denote as L_c^l. A higher weight for α means we place a higher value on the reconstruction of the original image. β is used to weight **style loss**, which we denote as L_s^l. A higher weight for β means we place a higher value on the transfer of the style from the style image to the content image.

$\mathcal{L}_{total}$ consists two summations. This is because we calculate a content loss for each convolutional layer we select for content representation and we calculate a style loss for each convolutional layer we select for the style representation. As a concrete example, for VGG-19, we use the 13^{th}, 14^{th}, and 15^{th} convolutional layers for content representations, and the 1^{st}, 2^{nd}, 3^{rd}, 4^{th}, and 5^{th} convolutional layers for style representations. So, L_c would be 3 and L_s would be 5.

To calculate a single $\mathcal{L}_c^l$, which is a content loss, we use equation:

$$\mathcal{L}_c^l = \frac{1}{H_l W_l C_l} \sum_{ij} (F_l[O] - F_l[I])_{ij}^2 \tag{2}$$

F_l refers to the output of the l^{th} convolutional layer after being run through the previous convolutional layers. This is a $H_l x W_l x C_l$ feature map.

So, this term is calculating the mean squared error of all pixels across all channels between a saved feature map, which in this case is from the original content image, and a feature map from the content image after being changed by optimization.

To calculate a single $\mathcal{L}_s^l$, which is a style loss, we use equation:

$$\mathcal{L}_s^l = \frac{1}{N_l^2} \sum_{ij} \frac{1}{H_l W_l C_l} (G_l[O] - G_l[S])_{ij}^2 \tag{3}$$

Each convolutional layer l has has N_l channels and D_l total pixels in each channel. $G_l \in \mathcal{R}^{N_l x N_l}$ is the gram matrix computed between $R_l(R_l)^T$ where $R_l \in R^{N_l x D_l}$ is the reshaped form of F_l.

So, this term is calculating the mean squared error of all pixels between the gram matrix of the reshaped feature map, which in this case is from the original style image, and the gram matrix of the reshaped feature map from the content image after being changed by optimization.

5.2 Total Variance Loss

One of the terms we find in the Lua implementation, but not in [6], is the total variance loss which is defined as:

$$\mathcal{L}_{TV} = \sum_{i=1}^{H} \sum_{m=1}^{W-1} |p_{i,m+1} - p_{i,m}| + \sum_{j=1}^{W} \sum_{n=1}^{H-1} |p_{j,n+1} - p_{j,n}| \tag{4}$$

Where $p_{i,j}$ refers to the value of the pixel in the i^{th} row and j^{th} column of the input image. In this case, the input image is the content image.
The loss is now:

$$\mathcal{L}_{total} = \sum_{l=1}^{L_c} \alpha \mathcal{L}_c^l + \sum_{l=1}^{L_s} \beta \mathcal{L}_s^l + \gamma \mathcal{L}_{TV} \tag{5}$$

γ is a hyper-parameter we use to weight the total variance loss. Intuitively, the total variance loss punishes sharp changes in pixel value and attempts to reduce "graininess".

5.3 Segmentation Loss

The baseline loss attempts to transfer the entire style of the style image to the content image without regard of similar subjects. For some instances, the user may want to constrain style transfer to similar subjects within both images.

In the baseline loss, the style loss is referred $\mathcal{L}_s^l$ for a convolutional layer l. Mimicking [6], we refer to the style loss with segmentation as $\mathcal{L}_{s+}^l$.

$$\mathcal{L}_{s+}^l = \sum_{c=1}^{C} \frac{1}{N_{l,c}^2} \sum_{ij} \frac{1}{H_l W_l C_l} (G_{l,c}[O] - G_{l,c}[S])_{ij}^2 \tag{6}$$

$$F_{l,c}[O] = F_l M_{l,c}[I] \tag{7}$$

$$F_{l,c}[S] = F_l M_{l,c}[I] \tag{8}$$

C refers to the number of classes we consider when segmenting. Previously, we refer to F_l as a feature map. For this augmented style loss, we must mask F_l C times and calculate the loss separately. To do this, we use a one-hot encoded mask $M_{l,c}$ which we preprocess before training and apply it to F_l to achieve $F_{l,c}$. The original mask is downsampled as we move down convolutional layers to keep the correct dimensions.

The loss is now:

$$\mathcal{L}_{total} = \sum_{l=1}^{L_c} \alpha \mathcal{L}_c^l + \sum_{l=1}^{L_s} \beta \mathcal{L}_{s+}^l + \gamma \mathcal{L}_{TV} \tag{9}$$

5.4 Matting Laplacian Loss

The process of separating the foreground object on an image from the background of the image is known as matting; the “matte” is the mask that can be used to retrieve the foreground object. Matting has been used in various applications in image and video processing, particularly to combine different images by placing the foreground object of one image onto another.

There is currently no way to characterize photorealism in a mathematical manner, but it is possible to maintain the existing photorealism of an image by preventing distortions during the style transfer process. We can do this by "creating a loss function based on locally affine transformations in color space that preserves edges in small patches" [10].

Work from Levin et. al showed how using a closed-form image matting method, which represents binary matte as an affine combination of RGB input channels, can help prevent distortions during the style transfer process [5] [10].

Let us denote $V_c[S]$ as the color channel c of the style image S and let us denote $\mathcal{C}_I$ as the matting laplacian matrix of the content image C. The affine loss is:

$$\mathcal{L}_m = \sum_{c=1}^{3} V_c[S]^T C_I V_c[S] \tag{10}$$

We expand Equation 1 above to include an extra term denoting the influence of this loss; the total loss is now:

$$\mathcal{L}_{total} = \sum_{l=1}^{L_c} \alpha \mathcal{L}_c^l + \sum_{l=1}^{L_s} \beta \mathcal{L}_{s+}^l + \lambda \mathcal{L}_m \tag{11}$$

λ serves as a hyperparameter to weight this loss, just as $gamma$ did in Equations 5 and 9.

5.5 Comparative Experiments

We qualitatively compare our style-transfer model with two popular and modern GANs, namely CycleGAN and MUNIT. Our style-transfer model is not a "typical" GAN in that it does not contain a discriminator and the style-transfer generator is only trained to over-fit with two specific images, so we are unable to make fair quantitative comparisons.

Cycle GAN The GAN framework has achieved impressive results in image generation. CycleGAN [9] leverages two GANs with a novel reconstruction loss to force the network to learn the data distribution in both image domains. Fig 2 shows the cycle GAN structure and its cycle-consistency loss.

Multimodal Unsupervised Image-to-Image Translation(MUNIT) Although CycleGAN generates promising results for style transfer task, it doesn't learn a disentangled representation of the image content and style. The MUNIT model solves this problem by learning the content latent space and style latent space separately. Figure 3 shows the basic architecture of MUNIT model.

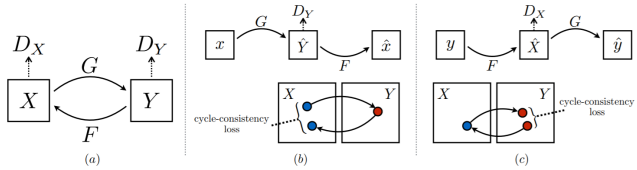


Figure 2: CycleGAN used for comparative experiments [9]

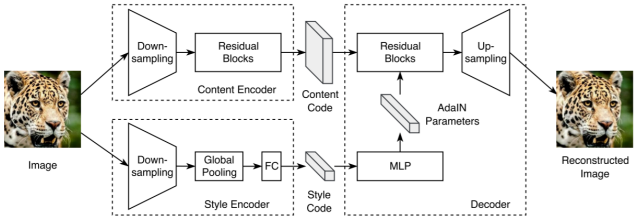


Figure 3: MUNIT architecture used for comparative experiments [9]

6 Model

6.1 Deep Neural Transfer

We will use PyTorch to implement a deep-learning approach leveraging a Generative Adversarial Network (GAN). For the feature extractor, we will use a pre-trained VGG-19 network [11] which has 16 convolutional, 3 fully connected, and 5 pooling layers. Although representations of the lower level layers reproduce the pixels of the original image, the representations of higher layers in the network can indicate objects and their arrangement within the image. These are called "content representations" [3].

We built feature spaces using the response from each layer of the network and we used these feature spaces to capture texture information to obtain a representation of an image's style. This information consists of filter responses and their corresponding spatial extent over the feature map. We can combine information from multiple layers of these feature spaces to obtain a stationary representation of the image that does not provide any information with regards to the arrangement of any particular object. These are known as "style representations" [3].

From VGG-19, we use 4^{th} convolutional layer as the content representations, and 1^{st}, 2^{nd}, 3^{rd}, 4^{th}, and 5^{th} convolutional layers as the style representations. To mix the content of a photograph with the style of another photo, we will minimize the distance between a white noise image from the content representation with the style representation of the reference photo in a number of layers of the CNN [3].

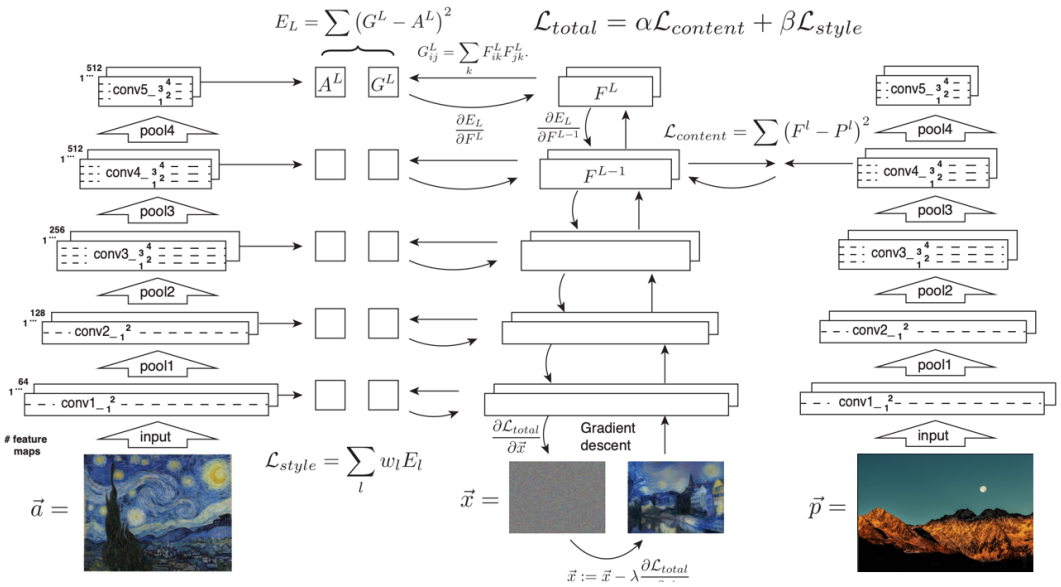


Figure 4: Style transfer algorithm. The model takes in a style image ($\vec{a}$) and stores its features at all five VGG19 convolutional layers (conv1, conv2, conv3, conv4, and conv5). The model also takes in a content image ($\vec{p}$) and stores the content image features after the fourth convolutional layer (conv4_2). These stored content losses ($L_{content}$) and the summed style losses (L_{style})combine via element-wise multiplication with tunable weights, α and β, respectively. This determines the overall loss (L_{total}) which the model works to minimize when generating a new image. To generate an image, the model passes an image of random pink noise ($\vec{x}$) with loss (G^L) through the network and calculates the total loss. It then takes the derivative of the loss and uses the gradient to back-propagate and update the generated image ($\vec{x}$) until the loss of the generated image matches the losses of the style image ($\vec{a}$) and the content image ($\vec{p}$). [2]

7 Results and Discussion

7.1 Baseline Model

We provide the loss in Figure 5 to show that the baseline model converges as expected. It is important to note that, since the style-transfer model is a "learning-free" model, it does not require a validation or test set. The model simply minimizes the difference between the loss of the generated image with the loss of the style and content images (see the model architecture 4 for more details). We do not provide graphs for other loss functions described in the Methods section because, similarly, they do not provide any relevant information. They only confirm that the model converges which is self-evident by looking at the generated images. Note that this is exactly what other style-transfer papers do, as well. Loss is not a very meaningful metric with this type of model.

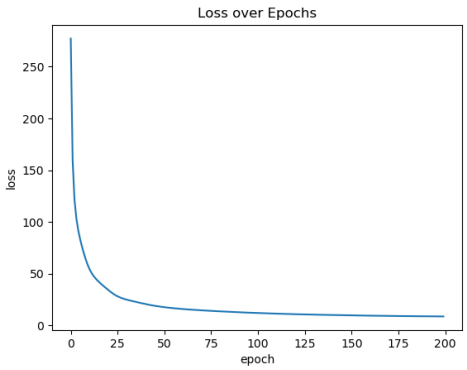


Figure 5: Baseline model loss.

7.2 Total Variance

Intuitively, the purpose of total variance loss is to reduce the amount of "graininess" in the image. We find that too high of a weight tends to blur the image, so a small γ such a 0.0001 to 0.00001 is usually used. The intended effect of removing graininess does work, but it also blurs some finer details. Overall, true variance loss does not seem to affect the transfer by a large amount and its uses may be up to user's preference. An example of the effects can be seen in Figure **6**.

7.3 Segmentation

When transferring style, the user may not want every single entity in the content image to be changed by the style. A comparison of a style transfer with and without segmentation can be seen in Figure **6**.

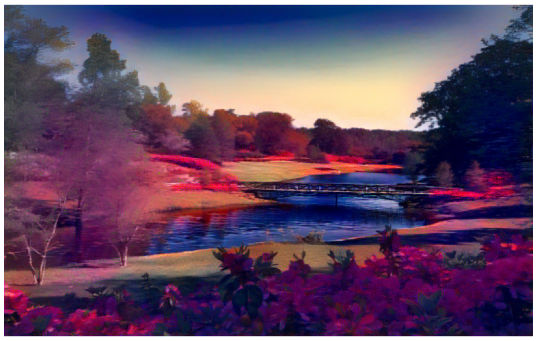
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

7.4 Matting Laplacian

We created a large sparse Laplacian matrix of size NxN, where N is the number of pixels in the image. We varied the λ hyperparameter and found that a value of 0.0001 was the best suited to produce images that more elegantly preserve straight edges in distinct parts of the content image. Figure 8 shows the result with and without the matting laplacian. It is evident that using matting produces a more photorealistic result, but the resulting image's colors are noticeably constrained. This is perhaps due to the color line constraint being slightly aggressive, but using smaller values of λ does not provide significant changes compared to not using matting.†g

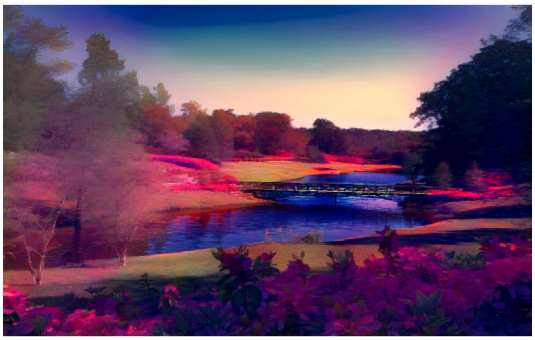
7.5 Original Colors

We implemented a simple color transfer function that superimposed the content image's original colors on top of the generated image. We did this by changing the color space from RGB to YUV. In a YUV color space the Y component determines the brightness of the color while the U and



(a) $\gamma = 0$

(b) $\gamma = 0.01$



(c) $\gamma = 0.0001$

Figure 6: Images with varying values of γ for total variance loss. (b) shows a value that is too high and (c) a value that works well with the image

V components determine the color itself. Once in the YUV color space, we replace the U and V components of the generated image with the U and V components from the original content image. This produces a generated image with the textures and styles of the style image with the colors of the content image. In Figure 9, the bottom left image shows the generated image with its colors unchanged and the bottom right image shows the generated image with the original colors of the content image.

7.6 Max Versus Average Pooling

By default, the pre-trained VGG19 model uses five max-pool layers. We found that replacing these layers with average pooling and increasing the style weights by a factor of 10^{-3} produced better blends of artistic style images. For example, in Figure 10, we combine Vincent van Gogh's famous *Starry Night* with a landscape photo. The right image is generated using the original model's max pooling layers we get shorter brush strokes and finer detail on the mountains. The left image is generated with a model that replaces the max pooling layers with average pooling layers. Here, we see longer brush strokes that better match the style of the original *Starry Night* painting.

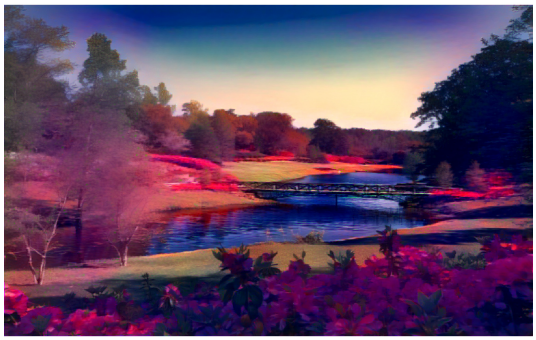
7.7 Experiments with Various Hyper-parameters

7.7.1 Alpha Versus Beta Tuning

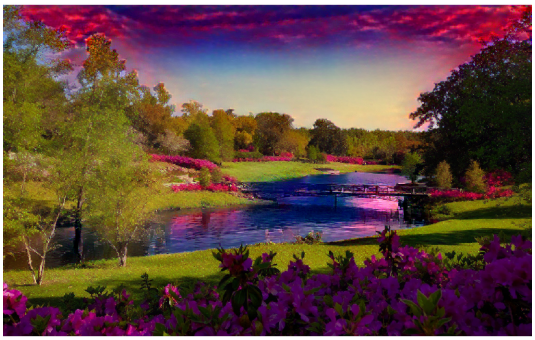
Generated images have tuneable parameters, alpha and beta, that control the prevalence of the content and style features in the generated image. Alpha represents the amount we weigh the loss of the content features and beta represents the amount we weight the loss of the style features. The more we weigh the style or content loss, the stronger those features come through in the generated image.

(a) Content Image

(b) Style Image



(c) Without segmentation



(d) With segmentation

Figure 7: (c) Shows an image without segmentation and (d) shows with. Both picture are "good" style transfers, but they are clearly different from each other. The only thing in common both pictures have is the sky and the water, which is why only the sky and the water have drastically different appearances.

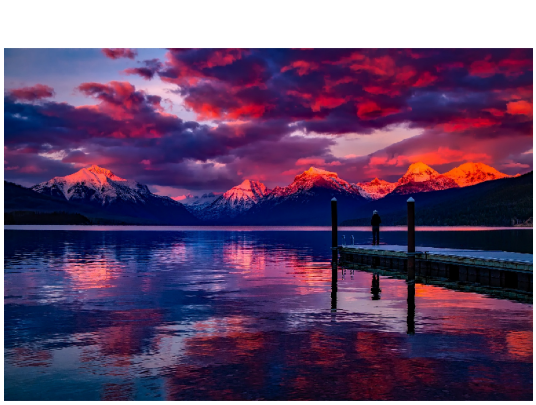
Because of their inverse relationship, we refer to the various settings of alpha and beta as a ratio. A high alpha/beta ratio (10^{-3}) encourages content features while a low alpha/beta ratio encourages style features (10^{-10}). Figure 11 combines Vincent van Gogh's *Starry Night* with a landscape photo. It illustrates the effects of different alpha/beta ratios. The top left starts with alpha/beta ratio of 10^{-3} and decreases from left to right, top to bottom, by a factor of 10. The bottom right is an alpha/beta ratio of 10^{-11}. We found the "sweet spot" to be and alpha/beta ratio between 10^{-4} and 10^{-8}. Images with ratios between 10^{-4} and 10^{-8} displayed a photo-accurate scene from the content image evenly blended with the color and texture of the style image.

7.7.2 Adam Versus LBFGS Optimization

We ran experiments using both Adam and LBFGS optimizers. We noticed that LBFGS experiments look longer to converge but ultimately produced better results compared to the Adam optimizer. Images generated with a LBFGS optimizer had more detail and better color accuracy. In Figure 12, the left image was generated using an Adam optimizer and the right image was generated using a LBFGS optimizer. With the image generated with LBFGS, the colors better match to content photo's colors and the image contains more detail.

7.8 Deep Style Transfer Versus MUNIT/CycleGAN

To compare the results, we use two datasets to train CycleGAN and MUNIT: 1, vangogh to landscape photos; 2. celebA to vangogh. The second dataset is intuitively much harder since human faces are more different from artistic landscapes. We make following interesting observations:
MUNIT performs good in collection photo style transfer where the two domains are all real photos,



(a) Content Image

(b) Style Image

(c) Without matting laplacian

(d) With matting laplacian, $\lambda = 0.0001$

Figure 8: (c) Shows an image without the matting laplacian and (d) shows with it; both results are relatively good, but it is clear that the image with the matting laplacian is much more photorealistic.

they fail to capture the style information when one domain is artworks. The generated images for both datasets have significant artifacts and are not desirable. Possible reason is that MUNIT makes the assumption that the two image domains share the same content latent space, which might not be the case when one domain is photos and the other is artworks.
CycleGAN can generate fairly good results for both datasets. But all group members agree that our generated images are more enjoyable, not to mention the fact that CycleGAN can't specify one image as the style.

7.9 Failed Model Variations

We also attempted a few other model variations that did not produce desirable results. We experimented with replacing the ReLU in VGG19 with a leaky ReLU thinking that the leaky ReLU, known for preventing "dead neurons," would improve the model. Adding the leaky ReLU did speed up convergence but it led to undesirable artifacts in the generated image. We hypothesize that the negative slope of the leaky ReLU is good for reducing loss but it creates non-desirable pixel values.

We also tried adapting a Wide ResNet50-2 model to the problem. Many newer and better performing models have been released since VGG19 and we originally hypothesized that we could get better results by using a more modern model such as the Wide ResNet50-2 model. We attempted mirroring the VGG19 content and style layer configuration in Wide ResNet50-2, but it yielded poor results. We tried numerous alternate style and content layer configurations and we also attempted modifying the alpha/beta ratio to try to improve the results and, again, we were unable to produce decent generated images. Unfortunately, exhausting the combinations of the number and location of both style and content layers in search for desirable results is prohibitively expensive both in terms of time and GPU resources.

Figure 9: Images comparing generated image with and without original colors function. Top left: style image, top right: content image, bottom left: generated image, bottom right: generated image with original colors.

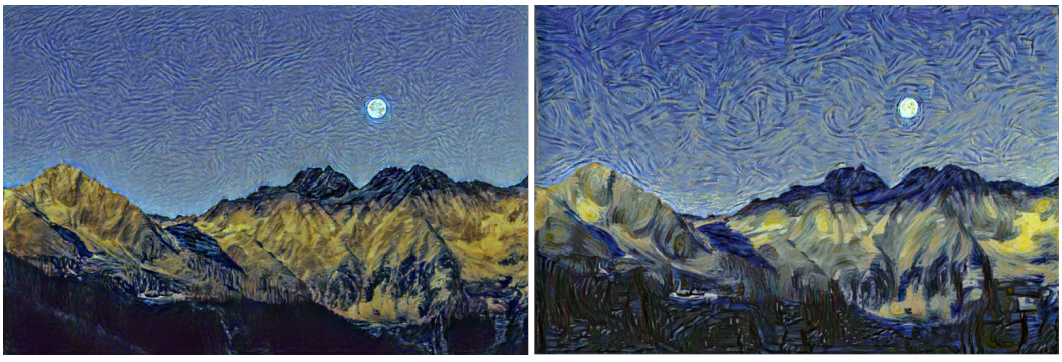


Figure 10: Images comparing max pooling with average pooling. Right: image generated with default max pooling layers. Left: image generated replacing max pooling layers with average pooling layers.

Figure 11: Visualized effects of various alpha/beta ratios. The top left is an alpha/beta ratio of $10{-}3$. The ratio decreases by an order of magnitude moving left to right and top to bottom across the image grid above. The final image in the bottom right has an alpha/beta ratio of 10^{-11}

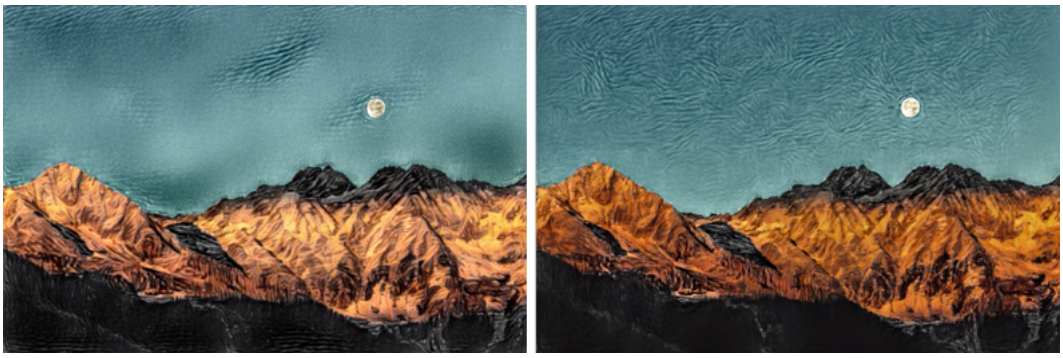
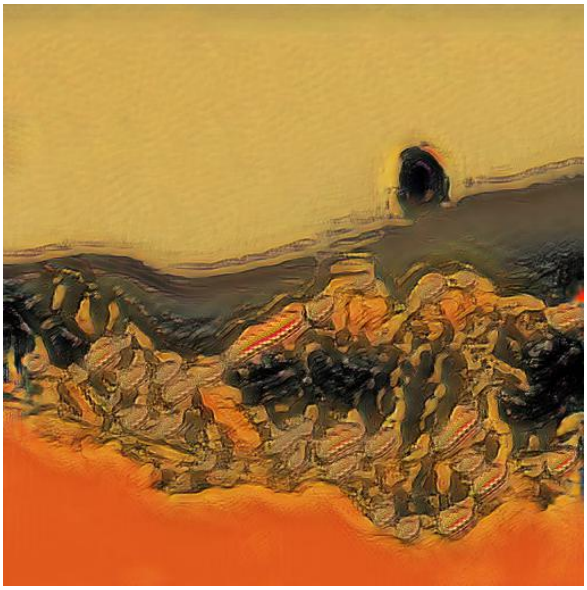
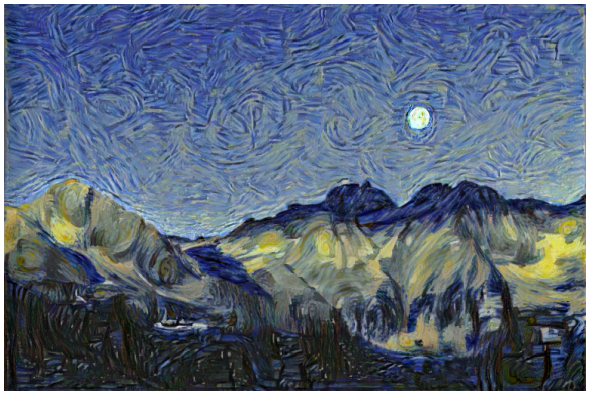


Figure 12: A comparison of the model running an Adam optimizer (left), and an LBFGS optimizer (right). We noticed the LBFGS optimizer generated sharper and more color-accurate images.



(a) MUNIT generated



(b) Style transfer network generated

Figure 13: MUNIT fails to capture the color information from artworks.

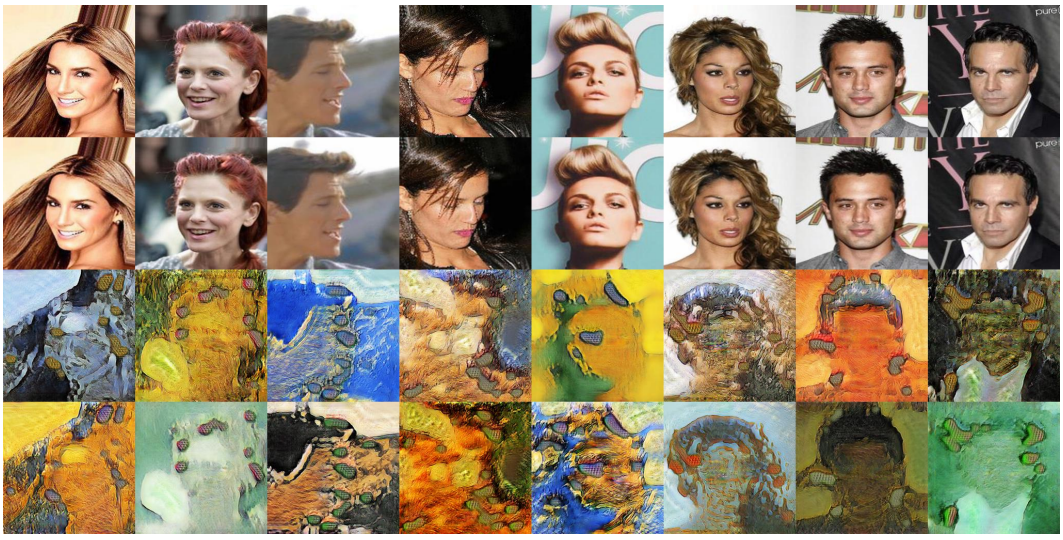


Figure 14: MUNIT fails to generate meaningful results for the celebA to Van Gogh task.

(a) Original Face (b) CycleGAN generated face

Figure 15: Cycle GAN generate better results than MUNIT, but it's still unrealistic.

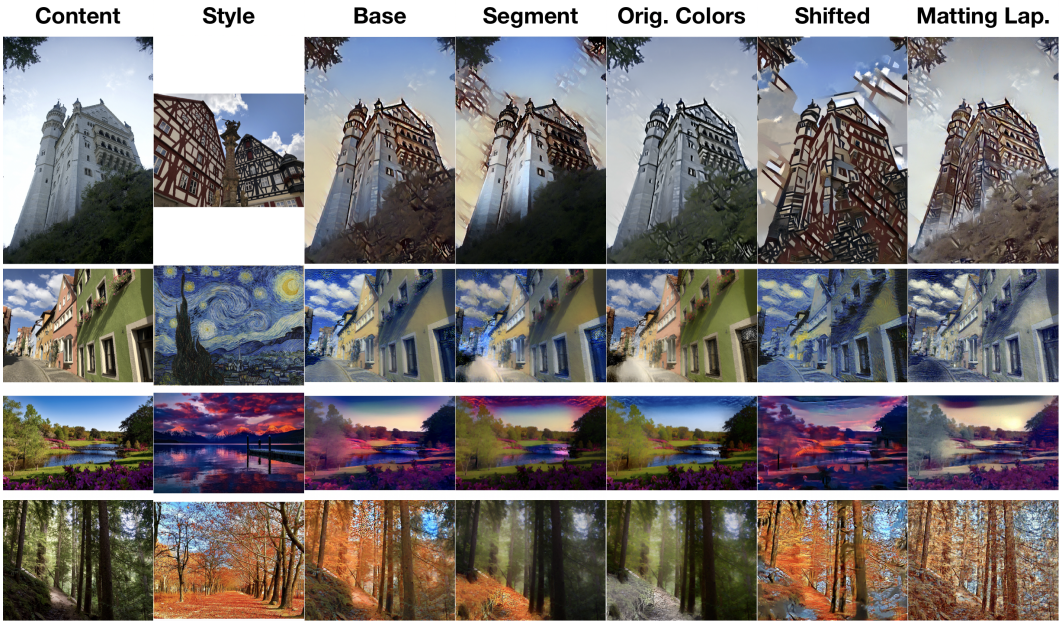


Figure 16: A side-by-side comparison between generated outputs of several different types of experiments.

Figure 17: Comparing the generated outputs from [9] to our best outputs.

After further reflection we hypothesize that, although there may be a combination of parameters that allow Wide ResNet50-2 to produce good generated images, Wide ResNet50-2's generated images will likely not be any better than the VGG19's generated images. Wide ResNet50-2 is considerably better than VGG19 at classifying ImageNet images (Wide ResNet50-2's 21.4 and 5.91 for top-1 and top-5 percent error, respectively, versus VGG19's 27.62 and 9.12 for top-1 and top-5 percent error, respectively), but that does not necessarily mean that it will be better at generating style transfer images. Generating style transfer images is not a classification problem—it is a problem of accurate feature extraction. Arguably, both the pre-trained VGG19 and Wide ResNet50-2 models are capable of extracting accurate features and, therefore, it does not follow that better classification models will generate better style transfer images.

8 Conclusion

In the process of replicating Luan et al's work, we tested some additional variants. We found that using segmentation to only apply the style to certain objects within the content image was generally successful in producing good results. Using the matting laplacian enabled us to generate results that were more photorealistic than without it. The CycleGAN and MUNIT models were effective in transferring a style onto a content image when the original images were similar, but they did not perform well when the style and content were from different domains. It is important to note that our base model is able to extract the style from one particular image, unlike the CycleGAN. The

MUNIT model has a key model assumption that prevents it from effectively transferring styles across domains.

Authors' Contributions

Cole's contributions

Cole ran experiments for the other two collection-style-transfer models: MUNIT and CycleGAN. He also wrote the Related Work and other corresponding parts.

Ramkesh's contributions

Ramkesh added the matting laplacian and wrote the related sections of the report and the conclusion. He also gathered images.

Vincent's contributions

Total variance loss, segmentation loss implementation, library for image segmentation, and corresponding parts in the report. Gathered landscape photos for data.

Varun's contributions

Unfortunately, Varun was very sick for the duration of the project. He was unable to significantly contribute because of his illness. He will be in touch with course admin regarding how to proceed.

Luke's contributions

Luke focused on generating and maintaining the quality of the final output images and animations. He ran various experiments comparing optimizers, architectures, and other hyper parameters. He also helped to gather images.

Bill's contributions

Bill built the base model and ran the following experiments: average versus max-pooling, original colors, alpha/beta ratios, Adam versus LBFGS, and the various experiments detailed in forsaken Section 7.9. He also wrote the corresponding sections in the report.

References

[1] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs, 2014.

[2] 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)*, pages 2414–2423, June 2016.

[3] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015.

[4] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. 2018.

[5] Anat Levin, Dani Lischinski, and Yair Weiss. A closed-form solution to natural image matting. *IEEE transactions on pattern analysis and machine intelligence*, 30(2):228–242, 2007.

[6] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. *CoRR*, abs/1604.04382, 2016.

[7] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 386–396. Curran Associates, Inc., 2017.

[8] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[9] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. Deep photo style transfer. *arXiv preprint arXiv:1703.07511*, 2017.

[10] Sebastian Penhouët and Paul Sanzenbacher. Automated deep photo style transfer. 2019.

[11] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[12] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networkss. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.