Two Methods For Style Transfer Using Pixel Art

Luke Fahmy
COGS 181 Spring 2020
lfahmy@ucsd.edu

Abstract

In this paper, I attempt and display the results of two different methods for applying the style of "pixel art" onto other images. This is motivated by the fact that neural style transfer tends not to work very well on pixel art and other art forms whose stroke style is more discrete than continuous. I use the original and most popular algorithm for neural style transfer [2] as a basis. I tweak the training process in two main ways and compare the results of those two ways with the original. The first way involves training with an ensemble of style images and weighting them according to importance. The second way involves trying to match the process of pixelization and depixelization to each other as done in [4]. The methods produce mixed results. However the results are interesting, and they provide many ideas for future work, especially on styles that are traditionally difficult for neural style transfer.

1 Introduction

Pixel art is a modern art form consisting of a discrete stroke style, where every stroke is a square of pixels, and each square usually has the same dimensions. To make pixel art based on some real image, artists must have an understanding of the semantic meaning of entities in the image in order to properly draw borders between entities using only squares of pixels. Additionally, since the pixels from an original image become clustered in the pixel art version, modern pixel arts tend to use a much smaller set of colors. This gives a cartoonish and animated effect, while still preserving meaning.

Of course, the original style transfer algorithm does not take semantic meaning into account. In addition, it does not preserve the colors of the content image, instead preferring the colors of the style image as discussed in [3]. These facts lead one to ask the question: What do we even mean when we say "style"? Does it involve stroke or texture or color or semantics, or all of the above? Take, for example, the three photos in Fig. 1 taken from [1]. Fig. 1(b) is a photo with the style of the famous "Starry Night" by van Gogh, but it clearly is not the true style. There are random circles of yellow taken from the stars in the painting, but if van Gogh were to paint this sleeping boy, he most likely would not have added these circles of yellow because there are no stars in the original photo. This leads us to conclude that Fig. 1(c) is a more true transferring of the style, because it keeps the colors of the photo, but it still uses strokes similar to the ones in "Starry Night". However, one can raise another concern. In Fig. 1(c) the strokes seem to be a meager attempt at copying the way van Gogh moves his brush, but they have no real purpose and carry no real meaning. In "Starry Night" the strokes change based on where they are in the painting, whether it be the hills or the sky. I have not answered the question about what we really mean by "style", and I am not trying to. I am only showing that the results of the most important algorithm for style transfer do not really satisfy our intuitive need to balance stroke, texture, color, and semantics.

In this paper, I experiment with two different methods to help address these concerns. The first method uses an ensemble of images of the pixel art style to try to make a natural use of the color of the content image. This method incorporates the loss between the content image and each individual pixel art image into the overall loss function. The second method tries to make the process of going

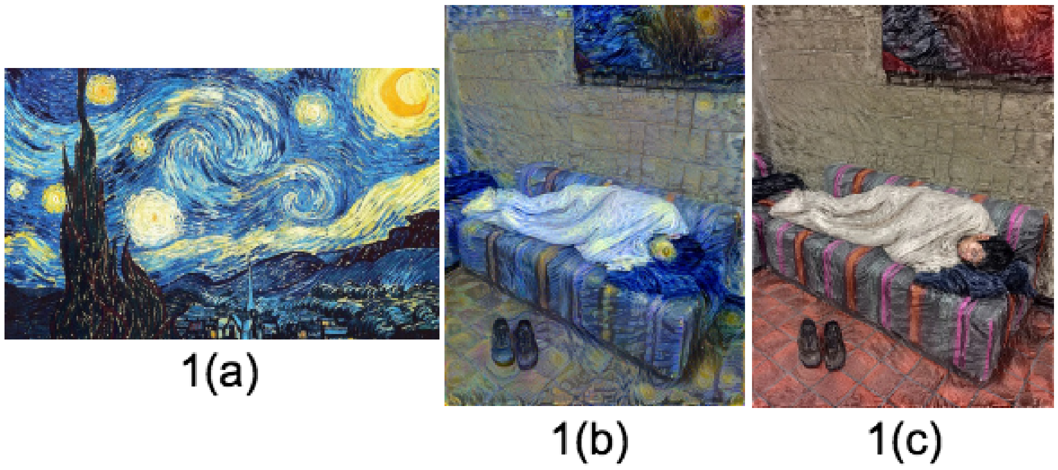


Figure 1: A style transfer with and without color.

from content to pixel art a reversible one, which might give the network a better overall understanding of the distinction between the two domains. It is inspired by [4]. The second method runs the content image through the original algorithm in [2] once, then runs that result through the original algorithm using the old content image as the style image and keeping track of the changing style along the way. Finally, it runs the original algorithm again using the original content and original style, but also trying to mirror the previous runs.

The first method produced interesting results and made some progress on the matter of automatic coloring, similar to [5]. The second method was severely misguided and would benefit from a different network architecture, but it nevertheless provided some insights and made some progress on the matter of color.

2 Methods

2.1 Ensemble Method

The original loss function in [2] is:

$$\mathcal{L}_{total}(p, a, x) = \alpha \mathcal{L}_{content}(p, x) + \beta \mathcal{L}_{style}(a, x) \tag{1}$$

where p is the content, a is the artwork, x is the image we are generating, α is the weight of the content, and β is the weight of the style. In our ensemble method, we only change $\mathcal{L}_{style}$. In [2], $\mathcal{L}_{style}$ is a linear combination of the mean-squared error between G^l and A^l, where G^l represents the Gram matrix of the pixel art in a given layer l, and A^l represents the Gram matrix of our changing image in l. Our new $\mathcal{L}_{style}$ does likewise except that it sums the mean-squared error between G^l_i and A^l where each G_i corresponds to one image in our set of style images. Since we factor in more than one image, we must weight each image somehow. Instead of weighting the images evenly, we weight images by how similar they are to the rest of the set. This is done by adding up the mean-squared error between the gram matrices of each image with every other image. The images with the highest MSE have the least weight in the loss function. With this method, the ratio of α and β stay the same, but multiple different pixel arts are given the opportunity to influence the result.

2.2 Gram Mirror Method

Suppose we have some content image p, and we clone it and try to add the pixel art style to it. Then suppose we take the resulting photo with pixel art style and try to add the style from the original content photo to it and get p'. Then we want $p \approx p'$, and this will mean our process is reversible. To do this, we run p through [2] algorithm and get an intermediate output i that is in the pixel art domain. Then we run i through the same algorithm and try to apply the style of p. For each iteration j in this second run of the algorithm and each layer l, we save G^l_j, which is the Gram matrix of i for layer l during iteration j. At the end of this second run, we have p'. Now we again run the original algorithm trying to apply the pixel art style to p. This time, however, we also try to minimize the difference between the Gram matrices of p and the Gram matrices of i. More formally, let A^l_k be the Gram matrix of p on iteration k in layer l. If we have n total iterations, then we want to minimize $(A^l_k - G^l_{n-k})^2$ for each layer l. In words, we are minimizing the MSE between the gram matrices of p and i. Since these processes are in reverse of each other, we must use the k^{th} Gram matrix of p and the $n - k^{th}$ Gram matrix of i. This has the effect of making the process more reversible, similar to [4]. For a specific iteration k, we define the mirror loss as $\mathcal{L}_{mirror} = \frac{1}{DHW}\sum_l \left(A^l_k - G^l_{n-k}\right)^2$, where D is the number of feature maps in a layer l, H and W are the height and width of each feature map, and n is the total number of iterations used to minimize the loss function. The total loss function becomes:

$$\mathcal{L}_{total}(p, a, x) = \alpha\mathcal{L}_{content}(p, x) + \beta\mathcal{L}_{style}(a, x) + \gamma\mathcal{L}_{mirror}(p, i) \tag{2}$$

3 Experiments and Results

3.1 Ensemble Experiments

For the ensemble set of pixel art, I found four reasonably different pixel artworks, and rotated them left, right, and down, for a total of sixteen images. It is important to keep the style set small, in order to minimize the amount of work required by the user, and keep it similar to other style transfer methods. Across all experiments with all methods, L-BFGS optimizer with a learning rate of .01 was the only way to get acceptable results. Also, for pixel art specifically, $\alpha = 1$ and $\beta = 1$ seem to be the best, although it is the ratio that is important.

3.2 Ensemble Results

The ensemble method offered clear improvements over the original algorithm. Edges are more clear and colors are closer to the original. However it still does not truly look like pixel art, and the side-by-side comparison in Fig. 2 makes that very obvious. It also results in some blur, which is one of the most difficult things to minimize, as discussed in [4]. Still, the result does look like an animated photo, and it is pleasing to look at.

3.3 Gram Mirror Experiments

For the gram mirror method, we can only choose one style image. The individual runs are exactly the same as the original algorithm, which is the same as running the ensemble method with only one image, so the optimal parameters remained about the same. One thing that affected the results in a non-trivial way was the coefficient γ from the loss function. A coefficient $\gamma = 1$ meant that the colors of the resulting image were very close to the original content image. A higher γ would lead the colors to be tinted in shades of the style image.

3.4 Gram Mirror Results

As shown in Fig. 3, the gram mirror method resulted in images with a lot of noise. However, this method seemed to clear up the edges in comparison to the ensemble method, while still making the image look animated to an extent. The process also produces three outputs as described above: i, p', and the final output. The three together show an interesting progression in the way the algorithm improved the result. Also, the ability to use γ to control the amount of color from the style image is something that I have not seen in any other paper/method.

4 Conclusion

In this paper, I explored two methods to improve the neural style transfer algorithm for pixel art. The ensemble method made no grand attempt to completely surpass the [2] method, so it consequently offered small improvements, especially in the color of the result. The gram mirror method sought to use the interesting ideas in [4] to achieve results similar to what they achieved. This attempt was severely misguided, even though it had interesting results. It would have benefited greatly from an actual network with trainable weights, instead of just VGG19 with no training. The only thing that experienced a gradient was the actual result image itself. This is why I had to use the unintuitive method of running through the network multiple times and saving some of the Gram matrices. In [4], they tried to minimize the difference between the corresponding feature maps that their pixelization and depixelization networks learned. This is a true mirror loss, and has weights that stay constant throughout one run through the network. In contrast, because I used a different gram matrix for each iteration, I essentially created a new loss function every iteration, which is very hard to optimize. In the future, I will consider a more stable architecture in addition to VGG19, such as a feed-forward neural network.

References

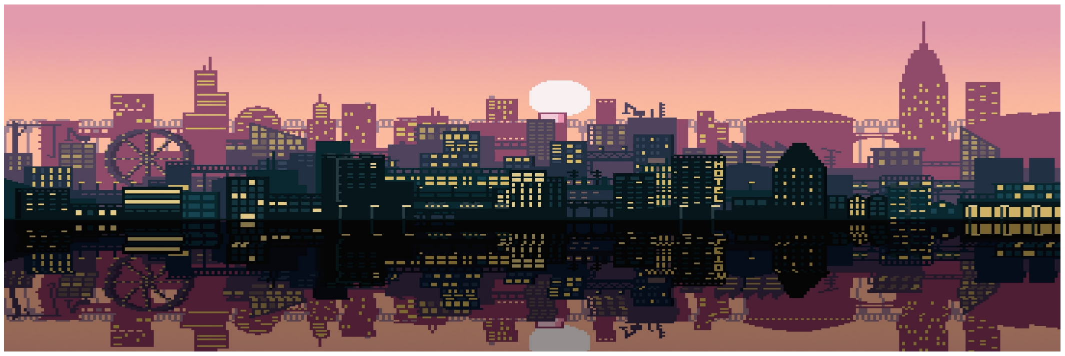
[1] Anish Athalye, . "Neural Style." https://github.com/anishathalye/neural-style. (2015).

[2] Gatys, Leon A., et al. "A neural algorithm of artistic style. arXiv 2015." *arXiv preprint arXiv:1508.06576* (2015).

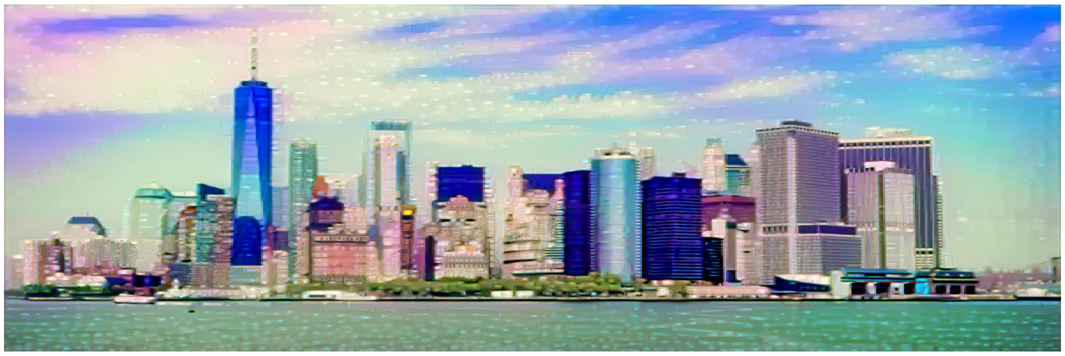
[3] Gatys, Leon A., et al. "Preserving color in neural artistic style transfer." *arXiv preprint arXiv:1606.05897* (2016).

[4] Han, Chu, et al. "Deep unsupervised pixelization." *ACM Transactions on Graphics (TOG)* 37.6 (2018): 1-11.

[5] Ikuta, Hikaru, Keisuke Ogaki, and Yuri Odagiri. "Blending texture features from multiple reference images for style transfer." *SIGGRAPH ASIA 2016 Technical Briefs*. 2016. 1-4.


(a) The original style image.

(b) The original content image.


(c) The result using the ensemble method.

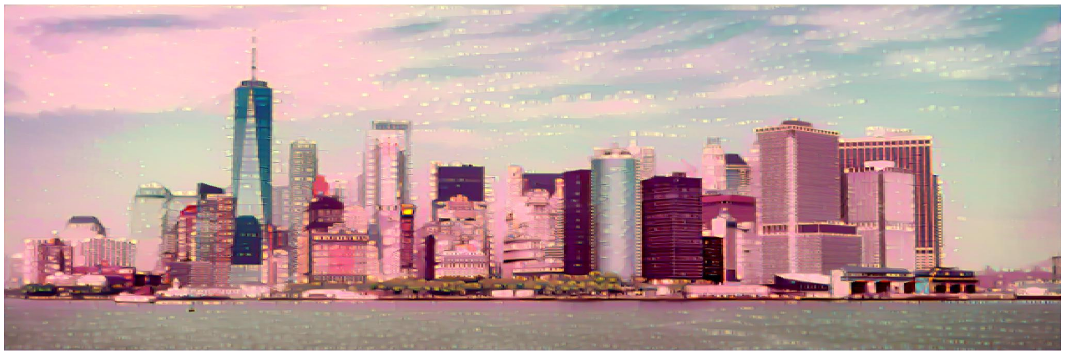
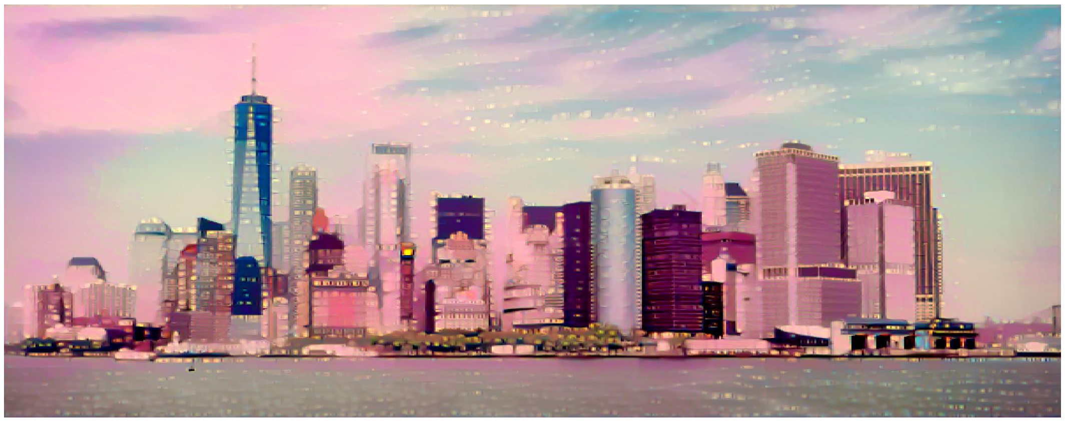
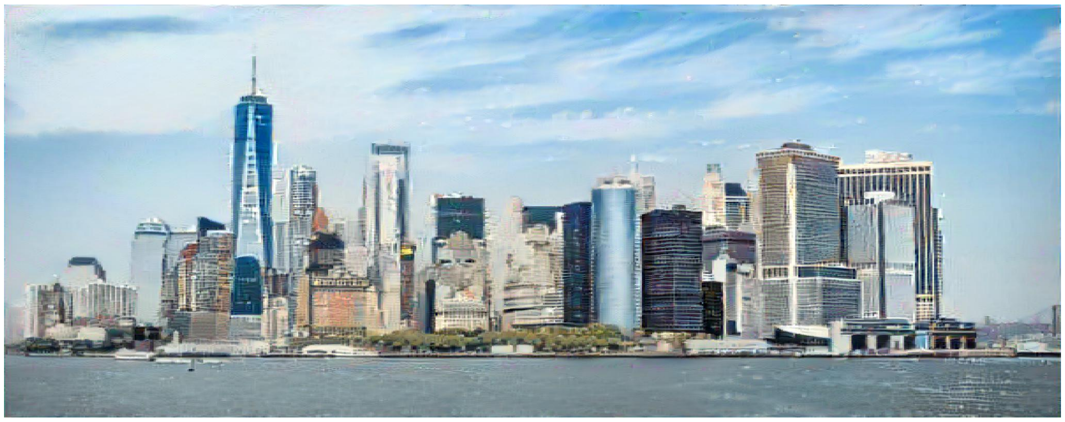

(d) The result using the original algorithm.

Figure 2: The ensemble method produces better color results without any manual or programmatic color correction.


(a) The result after one run through the method with 2(b) as the content and 2(a) as the style. Very similar to 2(d).


(b) The result after using 3(a) as the content and 2(b) as the style. We save the gram matrices of this run.

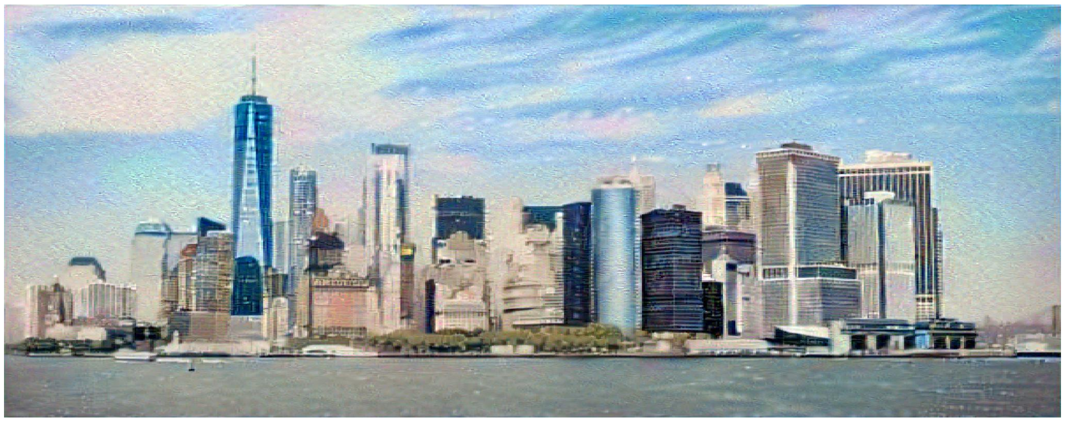

(c) The result after using the same content and style as 3(a) but with the mirror loss also.

Figure 3: The first, second, and final results from the gram mirror method.