Neural Style Transfer on Images

Huize Huang
Department of Statistics
Columbia University in the City of New York
hh2816@Columbia.edu

Fengyang Lin
Department of Statistics
Columbia University in the City of New York
fl2542@Columbia.edu

Yifei Xu
Department of Statistics
Columbia University in the City of New York
yx2577@Columbia.edu

Xiyao Yan
Department of Statistics
Columbia University in the City of New York
xy2431@Columbia.edu

1 Introduction

1.1 Goal

The goal for the project is to implement a style transfer algorithm, which is a part of texture transfer. In other words, the algorithm should be able to transfer the style from one image onto another, while preserving the content of the target image. Accordingly, the problem can be divided into three parts. The first task is to extract the semantic image content of the target image with those pre-trained Convolutional Neural Networks for object recognition. The next step is to extract the image style from the source image and obtain the style representation. Lastly, rendering the semantic content of the target image in the style of the source image will generate the new style-transferred image. Furthermore, we try different models and layers of neural networks to compare their performances on style transferring.

1.2 Related Work

Gatys et al. proposed A Neural Algorithm that use image representations derived from Convolutional Neural Networks optimized for object recognition. The algorithm allows people to separate and recombine the image content and style of natural images. [1,2] To reduce the run time, Johnson et al. then developed a real-time style transfer method with similar qualitative results but is three orders of magnitude faster. Their work also shows that with single-image super-resolution, replacing a per-pixel loss with a perceptual loss could provide visually pleasing results. [3] More recently, Dongdong et al. proposed StyleBank that composed of multiple convolution filter banks and each filter bank explicitly represents one style. [4]

Our implementation is mainly based on Gatys's work on Image Style Transfer. In this project, we use the same architecture of Neural Networks and loss function as in Gatys's work to implement our algorithm. [1,2] We also read Johnson and Dongdong's paper to gain a deeper understanding of the application and improvement of style image transfer and will implement their work if time permits. [3,4] TensorFlow's official tutorials on Neural style transfer and Victor's code on Kaggle serve as a starting point of our implementation. [5,6]

1.3 Data Description

Our task is not a typical machine learning problem. It is neither classification nor regression, thus we do not have to split the training/validation/test samples. In our task, there are two kinds of data, one is the target image (also referred as content image), the other is the style image (also referred as source image). Our goal is to extract "style", the painting features, from style image and then to

apply them to the target image. As a result, we could create a new image which integrates the content of target image and the visual properties of style image together.

For the basic part of our project, we want to apply different styles of great artists to the photos we took in our daily life. We will pick two content images and two style images (paintings, photographs, etc), and test our algorithm over the 4 resulting combinations of content and style. This is the main goal of our project.

Style images are all from a kaggle dataset, Best Artworks of All Time, a collection of paintings of the 50 most influential artists of all time. `https://www.kaggle.com/ikarus777/best-artworks-of-all-time` There are thousands of paintings in this dataset, we only picked the representative works from several world-renowned artists who have quite different styles, including Vincent van Gogh, Leonardo da Vinci, Pablo Picasso, Claude Monet, Salvador Dali, and Andy Warhol. Base images are collected from our phones, including the beautiful scenes in Columbia University and New York City, the lovely animals and delicious foods.

In our task, we also want to creatively apply the style of one artist to another artist, which may produce a strange result. In addition, we could also apply the style of one real photo to another photo. We believe that these are all interesting attempts.

2 Methods

2.1 Starting Point

The results of the original paper were generated on the basis of 19-layer VGG network. We started from reproducing the results with the setting in the original paper, i.e. the layers we pick from the VGG19 to extracts the contents and styles from the images and the hyperparameters including basic training iteration and loss loss weighting factors.

To be precise, we will implement the following steps and architecture as proposed in Gatys's paper.

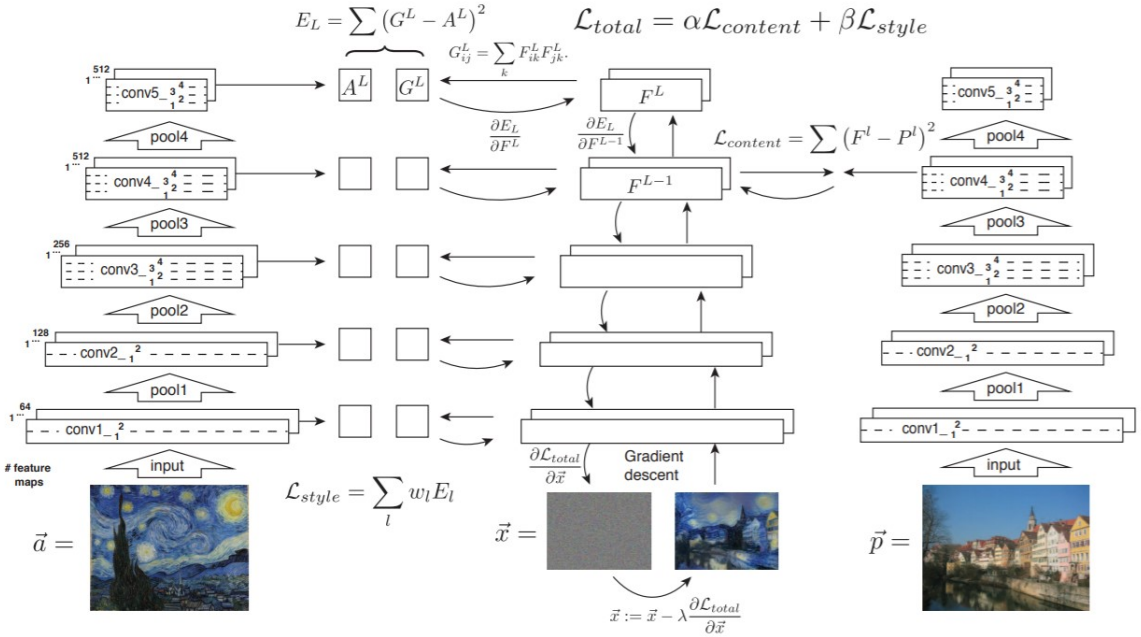

Figure 1: Style transfer algorithm from Gatys's paper.

- **Content Representation**

  To extract and store the content feature, the content image $\vec{p}$ and a randomly-generated white noise image $\vec{x}$ image are passed through the network. P^l and F^l are their respective

feature representation in layer l. We then calculated the squared-error loss between the two feature representations

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

where F_{ij}^l is the activation of the i^th filter at position j in layer l.

- **Style Representation**

  To extract and store the style feature, the style image $\vec{a}$is passed through the network. We use Gram matrix G^l to represent the style of the source image in layer l, where G_{ij}^l is the inner product between the vectorised feature maps i and j in layer l. Like what we did in content representation, we introduced $\vec{x}$ and minimise the mean-squared distance between the entries of the Gram matrices of $\vec{a}$ and $\vec{x}$. A^l and G^l are their respective stype representation in layer l. The contribution of layer l to the total loss is then

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_i^l j - A_i^l j)^2 \tag{2}$$

  and the total style loss is

$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l \tag{3}$$

  where w_l are weighting factors of contribution of each layer to the total loss.

- **Transfer Representation**

  The total loss is then a linear combination between the content and the style loss.

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}, l) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x}) \tag{4}$$

  where α and β are the weighting factors for content and style reconstruction, respectively.

Then we will compare our results with the results in the original paper in see if we reproduce its methods properly.

This is the first step of our project and it helps us understand the idea of Neural Style Transfer better and build the framework of our later work.

2.2 Algorithm Improvement (to do)

Once we build the framework, we will tune the following hyperparameters used in the algorithm.

- training iterations
- different random seeds to generate the white noise image
- the layers we picked from VGG19, we will try other layers in VGG19 and add or remove layers
- different weights on loss and different loss function

We will compare the output images from different models to find the most contributing factors and see if there are any interesting findings. If possible, we will also look at other pre-trained models and algorithms as mentioned in the introduction part like fast style transfer.

3 Results

The most important result of our task is the new image we created which combines the base image and style image. We will put the original content image, style image and new generated image together and check if the image we generated have the similar content with the content image, and the similar visual properties as style image. Since we do not have early results yet, we just show some sample results from Gatys's paper and kaggle at this time.

We would provide similar figures from our implementation in the final paper. We would compare a group of result images created by applying different style to one base image (like the left one in Figure 2), and another group of results images created by applying one style to different base images.

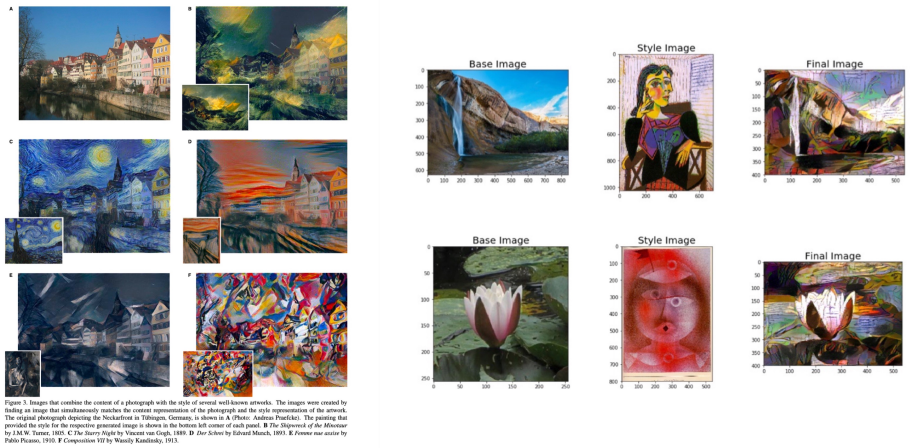

Figure 2: Sample result from Gatys's paper and kaggle.

In order to have a deeper understanding of what features each layer of neural networks has extracted, we may include the outputs of layers in the final paper (like the left one in Figure 3).

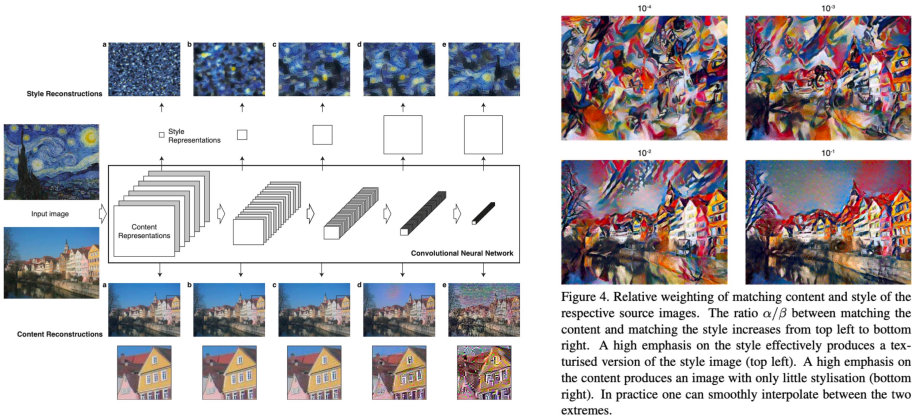

Figure 4. Relative weighting of matching content and style of the respective source images. The ratio α/β between matching the content and matching the style increases from top left to bottom right. A high emphasis on the style effectively produces a texturised version of the style image (top left). A high emphasis on the content produces an image with only little stylisation (bottom right). In practice one can smoothly interpolate between the two extremes.

Figure 3: a. Layer outputs b. Parameter tuning

We would also include the model details like model structures and hyperparameters. We may show the result images conducted by the models with different parameters (like the right one in Figure 3)

As for the model evaluation part, we would include the loss and running time.

4 Discussion

For us, this project helps us better understand how Neural Networks models extract different features of input (like content image and style image in this project) in different layers. When using the pre-trained model to do style transfer, it was found that lower layers simply taken the exact pixel values of the original image as features and then optimized. On the contrary, higher layers might capture high-level features of images and thus, give a better result when reconstructing the mixed output.

Similar inference can be applied to other applications of Neural Networks. In tasks like image recognition and localization, models with more layers usually have better results. This may lead to the classic discussion of how many layers are proper for a model.

However, we do think there is a limitation of the current algorithm. It's hard to tell which model or set of hyperparameters are better when tuning, for the loss may always have a great value and is not a good metric. Standard by people can be introduced, in our opinion, to decide the best model. We can invite people to choose the best one of the outputs from different models for one style, or to rate them. The best model can be chosen based on these results. This might not be the perfect metric for it is all about personal aesthetic standards, but still, it's another way to think about further work.

Last but not least, further research can focus more on real-time transfer mentioned in Johnson's paper, or using content video instead of content image to be the base. Every frame of the video can be seen as a content image, to which we can apply the algorithm. In addition, we can build a user-interface website which allows user to upload their own image. The algorithm can be used to help users to apply style filters to the uploaded photo.

References

[1] Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "Image style transfer using convolutional neural networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[2] Gatys, Leon, Alexander Ecker, and Matthias Bethge. "A Neural Algorithm of Artistic Style." Journal of Vision 16.12 (2016): 326. Crossref. Web.

[3] Johnson, Justin, Alexandre Alahi, and Li Fei-Fei. "Perceptual losses for real-time style transfer and super-resolution." European conference on computer vision. Springer, Cham, 2016.

[4] Chen, Dongdong, et al. "Stylebank: An explicit representation for neural image style transfer." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[5]"Neural Style Transfer nbsp;: nbsp; TensorFlow Core." TensorFlow, `www.tensorflow.org/tutorials/generative/style_transfer`.

[6]Basu, Victor. "Style Transfer Deep Learning Algorithm." Kaggle, Kaggle, 26 Apr. 2019, `www.kaggle.com/basu369victor/style-transfer-deep-learning-algorithm/notebook`.