Neural Style Transfer: Revisit and Improvement

Minyang Yu, Zimu Zhang, Zifan Wang
School of EECS, Peking University
{2200012739, 2200013115, 2200013080}@stu.pku.edu.cn

Abstract

Neural style transfer utilizes a neural algorithm to render a content image in the style of another image. In this article, we will show our implementation of the so-called Gatys-style transfer and Lapstyle transfer and compare their results. Besides, we extend these two methods by allowing for multiple style images, which is called multi-style transfer. Our method is based on weighting factors on style loss functions, which can achieve gradual change in different styles.

1. Introduction

The problem of style transfer has its origin from non-photo-realistic rendering in the field of *Computer Graphics*, which is closely related to texture synthesis and transfer. In our context, this problem could be formalized as:

> Given a **content image** A and a **style image** B (usually an artistic painting), try to generate a **stylised image** C that maintains the overall content of A but in the style of B.

Style transfer is important not only because it provides an easy way to generate new paintings in different styles, but also it helps us gain a deeper understanding of what "style" is and how "style" and "content" can be represented.

In this article, we mainly focus on methods involving neural networks to approach this problem. This kind of style transfer algorithms are hence called *neural style transfer*. In 2016, Gatys *et al*. [2] for the first time proposed to use convolution neural network (CNN) to tackle with this problem. We refer to their work as *Gatys-style*. Later Li *et al*. [7] improved Gatys-style transfer by adding the "Laplacian loss" term in the loss function. This method is referred to as *Lapstyle*.

Our work consists of three parts: (1) implement Gatys-style and Lapstyle transfer respectively. (2) compare the results of Gatys-style and Lapstyle transfer. (3) explore multi-style transfer.

The third part is our innovation. Multi-style transfer means there are more than one style image. We want to extract and combine the all these styles in the stylised image. For example, we want the style to be both Chines ink painting and Van Gogh's *Starry Night*.

2. Related Work

Some early approaches include histogram matching on linear filter responses [3] and non-parametric sampling [1, 6]. Since these methods typically rely on artificially designed low-level statistics, they often fail to capture high-level semantic structures.

2.1. Gatys-style Transfer

In 2016, Gatys *et al*. [2] demonstrated that image representations from deep convolution neural network (CNN) can somewhat separate the content and style of an image. Their work shows that different layers of a CNN extract information of different granularities. The style representation of an image can be obtained from the correlation of responses of different layers, which involves the *Gram matrix*. Then to generate an stylised image becomes an optimization problem, which requires to minimize both content difference and style difference. This could be achieved by gradient descent of neural networks.

2.2. Lapstyle Transfer

Later experiments and observations indicate that Gatys-style method leads to poor performance in maintaining the low-level structures (*e.g*. edges, contour segments, patterns, gradual color transitions) of the content image, and yields unappealing distorted structures and irregular artifacts in the stylised results.

To incorporate the detailed content structure, Li *et al*. [7] improved Gatys-style transfer by adding the "Laplacian loss" term in the loss function. The Laplacian of an image measures the magnitude of the gradient in every pixel. Hence, keeping similar Laplacian between the content image and stylised image helps to alleviate distortion. This designation proves to be effective and powerful which pre-

serves more details of the content image without bring harm to the quality.

2.3. New Architectures to Accelerate Operating Speed

Both Gatys-style and Lapstyle transfer try to optimize the stylised image through iterations, which takes a great amount of time even with modern GPUs. Then comes the workaround to replace the optimization process with a feed-forward network [5], which is about three orders of magnitude faster than the optimization-based alternative. Later Huang *et al*. [4] put forward the *adaptive instance normalization* layer to transfer arbitrary style to the content image with the encoder-decoder architecture. It also supports multi-style transfer.

Real-time style transfer is beyond our scope. Besides implementing Gatys-style and Lapstyle transfers, we focus on how to support multi-style transfer in this framework.

3. Data

Our data are all images which fall into 2 categories, namely, content images and style images. We have collected about 10 content images and 15 style images. Some of them are downloaded from the Internet, while others are from [2] [7]. Style images should be various because we need to verify if these neural style transfer algorithms are robust enough to perform well on different styles. Hence, our style images include abstract paintings, watercolor, cartoon drawing, sketches, *etc*.

In our implementation, all images are in "JPEG" format, with alpha channel neglected, since we only care about RGB values. The preprocess of data includes resizing content and style images to a given size, converting them to Pytorch tensors and align them to the given mean and variance.

4. Methods

4.1. Gatys-style Neural Style Transfer

In Gatys-style neural style transfer, Gatys *et al*. find the limitation that previous algorithms for style transferring only use low-level image features of the target image to inform the texture transfer. Therefore, Gatys *et al*. take advantage of Deep Convolutional Neural Networks (VGG-19 here) to extract high-level semantic information from natural images.

In VGG-19, with the number of layers increases, detailed pixel information gradually gets lost while the high-level content of the image is preserved. Besides, the correlation between different features in different layers, also known as the gram matrix, shows the texture information of an input image. So as the content image $\vec{p}$ passes through the net, from the reconstruction of the filter responses to an appropriate layer (conv4_2 here), we can get its content representation $P^l \in \mathbb{R}^{N_l \times M_l}$ (N_l is the number of feature maps, i.e. the channels number, and M_l equals the height times the width of the feature maps. We can get both N_l and M_l from layer l), which includes high-level content in terms of objects and their arrangement in the content image instead of exact pixel value of it. And as the style image $\vec{a}$ passes through the net, we can get its texture information A^l of different scales from a set of layers (conv1_1, conv2_1, conv3_1, conv4_1 and conv5_1 here). Similarly, when an input image $\vec{x}$ passes through the net, we can also obtain the content information F^l and texture information G^l of it. Here VGG-19 is just used as a feature extractor, but not trained on it.

The difference in the content information between $\vec{p}$ and $\vec{x}$ can be calculated by defining a loss function:

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

Also, the difference in the texture information between $\vec{a}$ and $\vec{x}$ can be calculated by summing up the averaged square difference of gram matrix in different scales:

$$L_{\text{style}}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l \left(\frac{1}{4{N_l}^2 {M_l}^2} \sum_{i,j} (G_{ij}^l - A_{ij}^l)^2\right) \tag{2}$$

where $\{w_l\}$ are weighting factors of the contribution of each layer to the style loss function. Thus, by minimising the total loss function

$$L_{\text{total}}(\vec{p}, \vec{a}, \vec{x}) = \alpha L_{\text{content}}(\vec{p}, \vec{x}) + \beta L_{\text{style}}(\vec{a}, \vec{x}), \tag{3}$$

we can get $\vec{x}$ which has similar content information response with the content image and similar texture information with the style image using gradient descent to adjust $\vec{x}$ according to $\dfrac{\partial L_{\text{total}}}{\partial \vec{x}}$. And the $\vec{x}$ with such content and texture information is just the content image in the style image's style.

4.2. Lapstyle Neural Style Transfer

Li *et al*. discover that in Gatys-style neural style transfer, the low-level features of the content image is absent while the low-level feature of style image dominates the detailed structures of the output image. However, in some tasks like style transfer of human, some regions such as faces and contours may be severely distorted, to which human perception is sensitive.

To solve the problem, in addition to the content loss and style loss of Gatys-style neural style transfer, Li *et al*. propose Laplacian Loss to steer the synthesized image towards having similar low-level structures as the content image, while being flexible enough to be rendered in the new

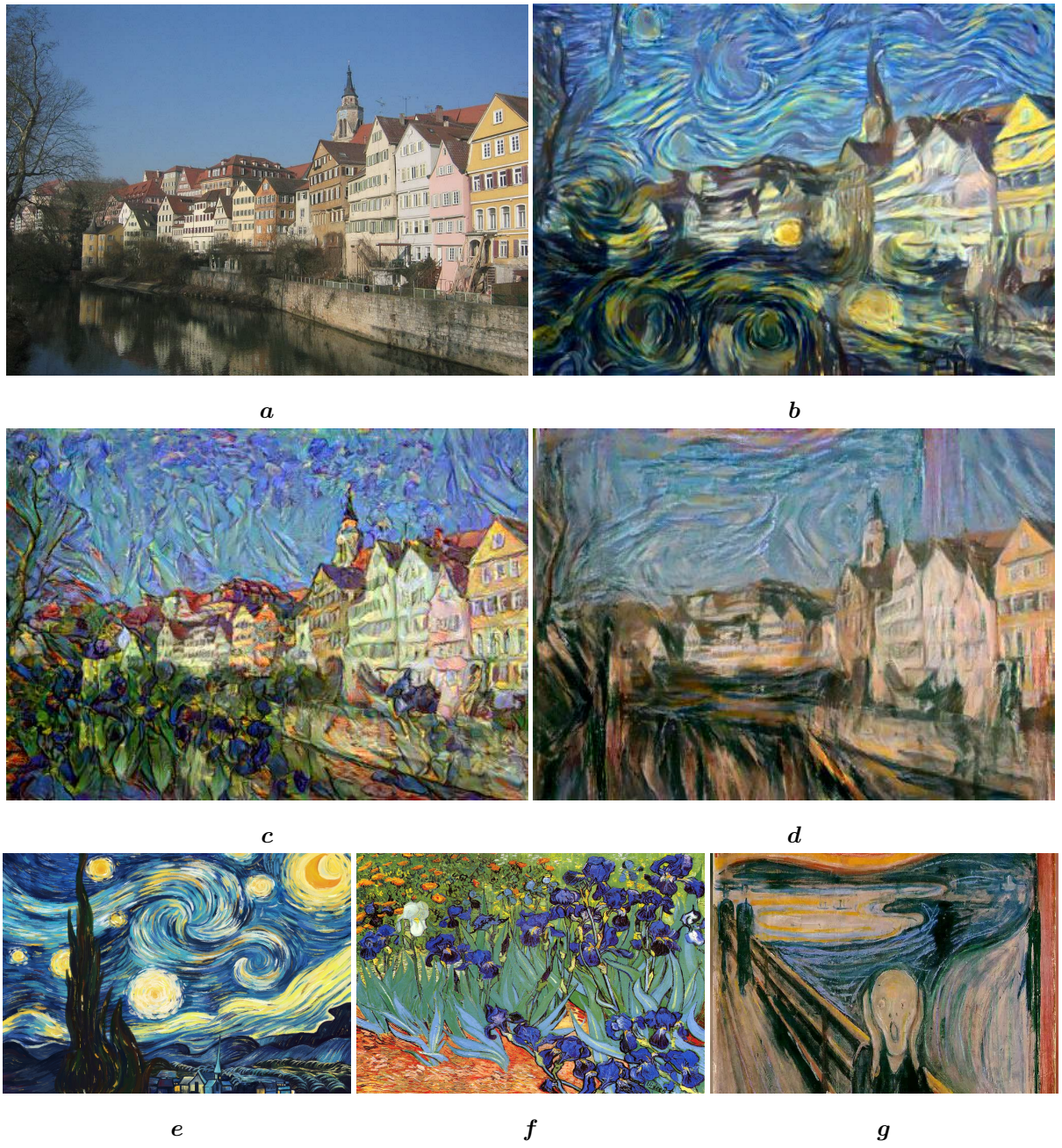


Figure 1: Gatys-style neural style transfer results with the style of several paintings.Picture $\boldsymbol{a}$ (Photo:Andreas Pracfcke) is the original image. Picture $\boldsymbol{b}$ is in the style of $\boldsymbol{e}$ (The Starry Night, Vincent van Gogh). Picture $\boldsymbol{c}$ is in the style of $\boldsymbol{f}$ (Irises, Vincent Van Gogh), Picture $\boldsymbol{d}$ is in the style of $\boldsymbol{g}$ (The Scream, Skrik).

style. To get the Laplacian loss, first we need to calculate the Laplacian matrix of an image $\vec{x}$. That could be done by convolving $\vec{x}$ with Laplacian filter

$$D = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix},$$

denoted by $D(\vec{x})$. For the RGB image with 3 channels, $D(\vec{x})$ is the sum of the absolute value of the Laplacian matrix for each channel, that is:

$$D(\vec{x}) = \|D(\vec{x}^R)\| + \|D(\vec{x}^G)\| + \|D(\vec{x}^B)\| \tag{4}$$

, but with some technical difficulties, Li *et al.* take

$$D(\vec{x}) = D(\vec{x}^R) + D(\vec{x}^G) + D(\vec{x}^B) \tag{5}$$

for simplification. The Laplacian loss then is defined as

$$L_{\text{lap}} = \sum_{i,j} (D(\vec{p}) - D(\vec{x}))_{ij}^2 \tag{6}$$

Besides, Li *et al*. add a $p \times p$ average pooling layer before the Laplacian convolution to make the filter less sensitive to small perturbations in the images as well as reduce the Laplacian loss to $1/p^2$ of that on the original images. Then the total loss of Lapstyle is the total loss of Gatys-style adds the Laplacian loss:

$$L_{\text{total}}(\vec{p}, \vec{a}, \vec{x}) = \alpha L_{\text{content}}(\vec{p}, \vec{x}) + \beta L_{\text{style}}(\vec{a}, \vec{x}) + \sum_k \gamma_k L_{\text{lap}_k}, \tag{7}$$

where L_{lap_k} is the Laplacian loss on the images pooled by an average pooling layer of size $p_k \times p_k$, and γ_k is its weight.

4.3. Our Improvement

In Gatys-style neural style transfer, we generally follow the ideas of Gatys *et al*. When it comes to Lapstyle, we make an adjustment that we use (4) to calculate $D(\vec{x})$ instead of the simplification (5). Besides, instead of adding an average pooling layer, we simply perform a *Gaussian Blur* to the content image before computing the Laplacian convolution, which can also eliminate noise on the content image and enable more accurate edge characterization.

4.4. Multi-style Transfer

To address the problem of multi-style transfer, we propose two approaches.

First, we come up with a very straightforward idea. Suppose we have two style images $\mathcal{S}_1, \mathcal{S}_2$, and one content image $\mathcal{C}$. We first perform a style transfer on $\mathcal{S}_1$ and $\mathcal{S}_2$ to get a mixed style image $\mathcal{S}_m$. Then we regard $\mathcal{S}_m$ as a new style image and perform style transfer once again with the content image $\mathcal{C}$.

Although this method is straightforward and easy to perform, there are a few drawbacks. If we have more than two style images, it will take a long time to blend those style images into one image. In addition, users can't set the ratio between those style images, which leads to the process uncontrollable.

According to those issues, we propose an improved approach. Based on the loss function in the *Lapstyle* method, we set a style loss term for each style images separately, and weight them all together.

The total style loss can be expressed as

$$L_{\text{style}} = \sum_i w_i L_{\text{style}}(\vec{a_i}, \vec{x}) \tag{8}$$

where $\{w_i\}$ are weights of those style images, which satisfy $\sum_i w_i = 1$.

5. Experiments

Our runnable code is available at GitHub[1], based on the code in *Dive into Deep Learning* [2]. In this section, we show the style transfer on various content and style images, using Gatys-style, Lapstyle and multi-style style transfer.

5.1. Gatys-style

Several new images are generated by Gatys-style style transfer in Fig. 1. We can see that the styles in style images are successfully transferred to the content image. The content information is preserved, while the low-level features are dominated by different texture information.

5.2. Lapstyle

Several new images are generated by Lapstyle transfer in Fig. 2. It can be discovered that although they use the same style images as those in Gatys-style style transfer, they do come up with different output images.

5.3. Comparision between Gatys-style and Lapstyle

A content image is transferred to the same style using the two different transfer method in Fig. 3.

First, it can be obviously seen that the one using Gatys-style contains more texture information than the one using Lapstyle, while the Lapstyle one keeps the figure of the boy better than the Gatys-style one. In an artistic way, Gatys-style is much better, but when it comes to legibility, the quality of Lapstyle will win.

Besides, as we can see in Fig. 4, for regions with more content information, like the body of the boy, they both have obvious difference. However, for regions with less content information such as the wall in the background, Gatys-style still has much difference to the original image, while that difference of Lapstyle decreases a lot in those areas.

The reasons for them both lie in the Laplacian loss used by Lapstyle. It inhibits the process where low-level content information is dominated by the texture information of style image, so visually, the one using Lapstyle will keep more features of content image. Also, for the areas with pure color (like the white wall), the value of $D(\vec{p})$ is close to zero, so the increase in texture information in such areas will result in rapid rise in the value of Laplacian loss, thus the Lapstyle with Laplacian loss will set less change in the areas. That can explain why the background of Gatys-style looks more colorful.

5.4. Multi-style

As section 4.4 illustrates, we first try to blend two different style images to generate a mixed style image. Using the

[1] https://github.com/flaricy/Image-Style-Transfer

[2] https://d2l.ai/chapter_computer-vision/neural-style.html

a ***b***

c ***d***

Figure 2: Lapstyle neural style transfer results with the style of several paintings.Picture ***a*** (Photo:Andreas Pracfcke) is the original image. The style image of ***b***, ***c***, ***d*** is in the same sequence of that in 1

a ***b*** ***c***

Figure 3: Picture ***a*** is the original image. ***b*** uses Gatys-style neural style transfer, while ***c*** uses Lapstyle neural style transfer

mixed style image, we perform another style transfer on the content image. The results look good in some cases (Fig. 5).

As for the improved approach we proposed, we pick the pku snow as the content image and choose starry and Kandinsky as the style images. In order to test the blend consecutiveness of our method, we try different weight combinations (Fig. 6). As we can see in the figure, if we make the starry style proportionally larger, there will be more circular stripes appearing in the result image. On

$\boldsymbol{a}$ $\boldsymbol{b}$ $\boldsymbol{c}$

Figure 4: Picture $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$ is the Laplacian matrix of 3 in the same order

Style 1: sketch

Style 2: Kandinsky

Mixed style: sketch-Kandinsky

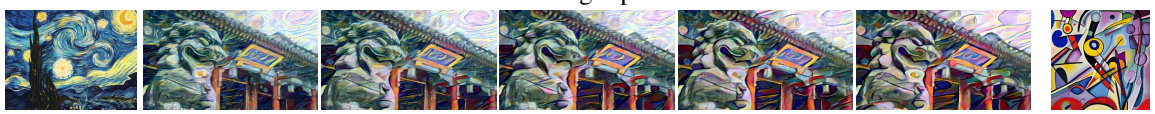
Content image: Alps

Final image

Figure 5: We first blend the two style images to get the mixed style image. Then we perform the style transfer once again on the content image and mixed style, and finally obtain the final image.

the other hand, if we make the Kandinsky style proportionally larger, more tangible geometrical shapes will show up in the result image, while color in the result image will be more vivid as well.

Content image: pku snow



Starry 9:1 7:3 5:5 3:7 1:9 Kandinsky

Figure 6: Here we have one content image, pku snow, and two style images, starry and Kandinsky. By adjusting the blend ratio, we can make a bunch of different blend results.

6. Conclusion

We have implemented Gatys-style and Lapstyle transfer and test them on various images, which proves to perform well and generate some beautiful images by elaborately tuning hyper-parameters. By comparing the results of Gatys-style and Lapstyle transfer, we get conscious that Lapstyle preserves more low-level structures of the content image in the stylised image, while what Gatys-style generates looks vague in local areas (Fig. 3). In art, however, details may not be the top priority. Since Gatys-style transfers more "style" to some extent (by comparing Fig. 1 with Fig. 2), one cannot judge that Lapstyle transfer is absolutely better than Gatys-style.

We also explore multi-style transfer. Applying different weighting factors to the style loss of each style image yields different stylised image (Fig. 6).

Frankly speaking, hyper-parameters in Gatys-style and Lapstyle transfer are hard to tune well. As for the optimizer in neural networks, our experiment, altogether with other papers, comes up with the conclusion that L-BFGS is better than Adam, which is more frequently used though. Future research may explore the reason behind.

References

[1] A. Efros and W. Freeman. Image quilting for texture sunthesis and transfer, 1999. In ICCV.

[2] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks, 2016. Proceedings of the IEEE conference on computer vision and pattern recognition.

[3] D. Heeger and J. Bergen. Pyramid-based texture analysis/synthesis, 1995. In SIGGRAPH.

[4] Huang, Xun, and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization, 2017. Proceedings of the IEEE international conference on computer vision.

[5] Johnson, Justin, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution, 2016. Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14. Springer International Publishing.

[6] O.Frigo, N. Sabater, J. Delon, and P. Hellier. Split and match: example-based adaptive patch sampling for unsupervised style transfer, 2016. In CVPR.

[7] L. Shaohua and et al. Laplacian-steered neural style transfer, 2017. Proceedings of the 25th ACM international conference on Multimedia.