Neural Style Transfer

Zehua Chen, Jiaying Yang, Yiping Xie (each member contribution level is 3)

DD2424 Deep Learning in Data Science, KTH Royal Institute of Technology

Abstract. Neural style transfer is a deep neural system to synthesis a new image that obtains the high-level information of a content image yet rendered with the texture of a style image. The pre-trained convolutional neural network (CNN) VGG-19 is now frequently used in this topic, because of its powerful performance in separating and recombining content and style of arbitrary images. In addition to the ordinary neural style transfer, we also realize some further functions, including multi-style transfer and color-preserving transfer. What is more, we also try out some improvements in the calculation of loss function by adding total variation and Laplacian term into it, which help to get the output images with better quality.

Keywords: Style Transfer, convolutional neural network, VGG-19, Laplacian loss

1 Introduction

We have upload all of our code in this project on Github[1].

Neural style transfer (NST) is a technique to generate new image that can replicate the content of one input image, but combined the style of another one. This technique can be used into changing an ordinary photography into artistic style, by giving an input of an arbitrary art work. The work of Gatys *et al.*[1] first demonstrated the power of Convolutional Neural Networks (CNN) in neural style transfer.

In this project we first replicate the experiment done by Gatys *et al.* to realize the neural style transfer. Their used Gram matrix to realize the reconstruction of image style, which indicates the correlations between filter responses in different layers of a pre-trained CNN classification network, namely VGG-19.

Then we realize some more functions based on the theory of neural style transfer, including multi-style transfer and color preserving transfer. Multi-style transfer can transfer the combination of several style into the content image , while color preserving transfer can keep the original color of the content image unchanged after transfer.

Finally, we made some further improvements to the calculation of loss function to improve the quality of the generated image. We add the total variation regularization term to the loss function in order to improve the smoothness of

[1] https://github.com/Chen-Z-H/Neural-Style-Transfer-based-on-CNN

the output image. Moreover, we add a Laplacian loss term to the total loss function in order to reduce the artifacts and distortions of the generated image by taking the pixel space into account.

2 Background

After Gatys *et al.* proposed to use CNN to render a content image in the style of famous artworks, numerous studies followed up to either extend or improve the algorithm. The NST algorithm can be seen as the combination of visual texture modelling (and synthesis) and image reconstruction.

Based on how the visual texture is modeled, there are mainly two distinct approaches: Parametric Texture Modelling and Non-parametric Texture Modelling.

The parametric visual texture approach Gatys *et al.*[1] proposed is to design a Gram matrix, which calculates the correlations between the CNN filter responses in different layers of VGG network, to measure the second-order statistics of the filter responses in the CNN domain. This Gram-based texture representation in the CNN domain effectively models a variety of natural and non-natural textures.

Another visual texture modelling method is non-parametric, most of which is based on Markov Random Fields (MRFs) model. This model assumes that each pixel in a texture image is fully characterised by its spatial neighborhood. Wei and Levoy[2] proposed an approach to synthesis each pixel by searching a fixed neighborhood in the texture example image. One of the critical limitation of MRFs is that estimating the distribution of image patches from the example data is difficult and another limitation MRFs suffer from is sometimes local image statistics have a hard time fully capturing the texture of the complex image. However, this patch-level matching algorithm can preserve fine structure and arrangement better, which means it can work remarkably for photo-realistic styles.

Based on the speed of image reconstruction, there are also two approaches: Slow Image Reconstruction and Fast Image Reconstruction.

The idea based on Slow Image Reconstruction is to model and extract the content and style information and then recombine them as the target representation, which can be updated iteratively until it matches the desired CNN feature distribution. This optimising process can be very time-consuming and inefficient thus lots of researchers proposed an approach to leave the huge computation burden at the training stage: training a feed-forward neural network in advance so that the reconstruction process can be done simply with a forward pass at the test stage[3][4]. These Fast NST algorithms are able to generate stylised images in real-time, even though in terms of minimising the loss function, they are not good as Slow NST algorithm[5]. Depending on the number of styles one model can produce, Fast NST algorithms can be divided into Per-Style-Per-Model[3][4], Multiple-Style-Per-Model[6][7], and Arbitrary-Style-Per-Model[8][9].

3 Approach

3.1 Theory

The theory below mainly follows Gatys *et al.*'s work.

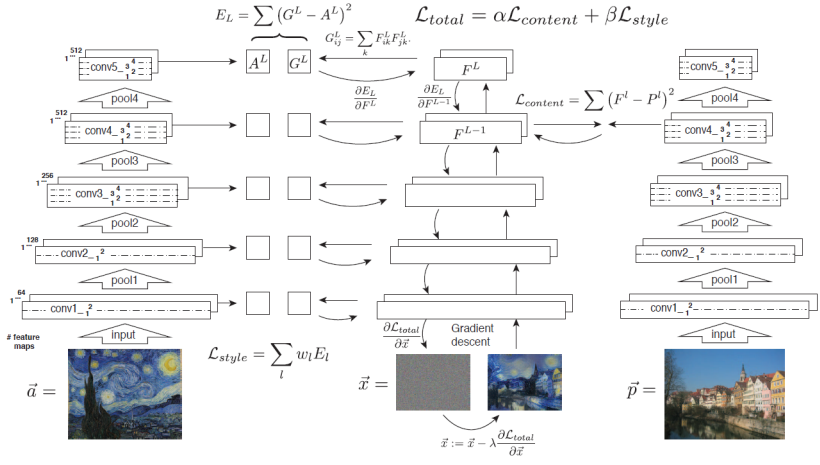


Fig. 1. Basic style transfer algorithm

In the VGG network, each layer represents a filter response to the image. Given a source image $\boldsymbol{x}$, a layer l with N_l distinct filters has N_l feature maps each with the size of M_l, where M_l equals to the multiplication of the feature map's height and width. Thus the response for layer l can be represented as a matrix $F^l \in \mathcal{R}^{N_l \times M_l}$ where F^l_{ij} is the activation of the i_{th} filter at the position j in layer l. Assume $\boldsymbol{p}$ and $\boldsymbol{x}$ to be the original content image and the generated image respectively, and F^l and P^l to be the respective feature representations in layer l. Then the content loss can be described as squared-error loss between these two representations:

$$L_{content}(\boldsymbol{p}, \boldsymbol{x}, l) = \frac{1}{2} \sum_{i,j} (F^l_{ij} - P^l_{ij})^2 \tag{1}$$

Here a parametric texture modelling is used to capture the visual texture information of the input style image. This feature can be calculated as the correlations between the different filter responses in ant layer of the VGG network, given in the format of Gram matrix:

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk} \tag{2}$$

Assume $\boldsymbol{a}$ and $\boldsymbol{x}$ to be the original style image and the generated image respectively, and A^l and G^l to be the respective style representations in layer l. The style loss at each layer is given by:

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_{ij}^l - A_{ij}^l)^2 \tag{3}$$

The total style loss is the combination of losses at different layers:

$$L_{style}(\boldsymbol{a}, \boldsymbol{x}) = \sum_{l=0}^{L} w_l E_l \tag{4}$$

Our aim is minimizing the distance of the synthetic image from the content representation as well as the style representation given by the formulas above. Thus we combine the style loss and content loss in following way:

$$L_{total}(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{x}) = \alpha L_{content}(\boldsymbol{p}, \boldsymbol{x}) + \beta L_{style}(\boldsymbol{a}, \boldsymbol{x}) \tag{5}$$

3.2 Further Functions

Multi-style Transfer We can intuitively understand that the style transfer algorithm mentioned above is actually a optimization strategy that balancing the loss between multiple images. Then it is easy to come to an idea that if we balance the losses between multiple styles, we can combine the styles of several art works to generate the image. This can be achieved by modifying the formula as below:

$$L_{multi} = \sum_{i=1}^{n} w_i L_{style}(\boldsymbol{x}, \boldsymbol{p}_i) \tag{6}$$

where $p_1, p_2, .., p_n$ is a series of style images and w_i is the weight of style loss of each image. In this paper we perform the experiments of blending two distinct painting styles but we can also extend it to a arbitrary number of styles. By changing the weights of style losses we can adjust the effect each style image poses on the final generated image.

Color Preserving Style Transfer When using the traditional algorithm to transform an content image to have the style of another given style image, we always face the shortcoming that the color of the content image is also changed. The unexpected altering of color is what we hope to avoid. Thus we improve the algorithm in order to realize the function of Color Preserving Style Transfer.

As is shown in the article[10], at present there are two main methods to realize the function of color preserving, which are Color Histogram Matching and Luminance-only Transfer. We choose to use Luminance-only transfer, because compared to the first method, it can preserve the colors of the content image better.

Luminance-only transfer is realized by doing the style transfer only in the luminance channel. We first transfer the content image and style image from RGB space into YUV space. In YUV space, the Y channel stands for the luminance information, while the U and V channels represents the color information. In order to preserve the color information of the content image and only change its luminance information, we should extract the Y channel from the style image, and the U and V channels from the content image. Finally the extracted three channels are regrouped to produce our final output image. This process is shown in Fig.2.

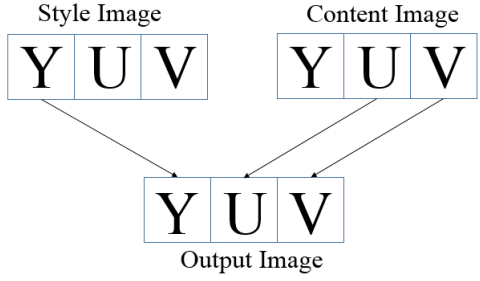


Fig. 2. Luminance-only Transfer

In order to improve the performance of Luminance-only Transfer further, we transform the luminance of the style image L_S to have the same first and second order statistics as the content image L_C as below:

$$L_{S'} = \frac{\sigma_C}{\sigma_S}(L_S - \mu_S) + \mu_C \tag{7}$$

where L_S and L_C are the luminance information of the style and content images; μ_S and μ_C are the mean luminances of the two images; and σ_S and σ_C are their standard deviations.

3.3 Total Variation Regularization

When calculating the total loss, in addition to the content loss and style mentioned in Equation 5 , we can also add a total variation regularization term to the total loss L_{TV}[3], which can help encourage spatial smoothness in the final output image.

The total variation is the sum of the absolute differences for neighboring pixel-values in the input images, which measures how much noise is in the images. After adding the total variation into the calculation of loss function, it can help suppress noise in the content image. The calculation of total variation loss L_{TV} is calculated by using the Tensorflow function $tf.image.total\_variation$.

The calculation of loss is updated as below after this improvement:

$$L_{total}(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{x}) = \alpha L_{content}(\boldsymbol{p}, \boldsymbol{x}) + \beta L_{style}(\boldsymbol{a}, \boldsymbol{x}) + \theta L_{TV}(\boldsymbol{p}) \tag{8}$$

where θ is a constant parameter controlling the strength of the total variation regularization.

3.4 Laplacian-Steered Neural Style Transfer

In the algorithm mentioned above, the content loss is computed in the CNN feature space so that the synthetic image is semantically similar to the content image. However, the CNN features are high-level representations that mainly capture the semantic information, thus inevitably leading to lose some low-level features of the image, e.g. texture, colors and basic structures. The synthetic image will usually have some irregular artifacts and unsatisfied distortions[5].

To reduce the artifacts and better preserve the detailed structures of the content image, a new term named Laplacian Loss is added in the loss function[11] by measuring the squared Euclidean distance between the Laplacian operator response of the content image and the synthetic image.

Given an image $\boldsymbol{x}$ the Laplacian filter response $D(\boldsymbol{x})$ is computed by applying the two dimensional discrete Laplacian Operator D to the image.

$$D = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{9}$$

Given the content image $\boldsymbol{p}$ and the synthetic image $\boldsymbol{x}$, the Laplacian loss L_{lap} is defined as

$$L_{lap}(\boldsymbol{p}, \boldsymbol{x}) = \sum_{ij} [D(\boldsymbol{p}) - D(\boldsymbol{x})]^2_{ij} \tag{10}$$

Thus the total loss is defined as:

$$L_{total}(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{x}) = \alpha L_{content}(\boldsymbol{p}, \boldsymbol{x}) + \beta L_{style}(\boldsymbol{a}, \boldsymbol{x}) + \gamma L_{lap}(\boldsymbol{p}, \boldsymbol{x}) \tag{11}$$

where γ is a tunable parameter controlling the strength of the Laplacaian Loss constraint.

There are several practical details during the calculation of the Laplacian loss[11] that can be addressed as below.

Laplacians on RGB channels Since the input image $\boldsymbol{x}$ has RGB channels, the Laplacians should contain the result on each channels separately. Regardless the sign, the Laplacian operator can detect the edge when there is a huge magnitude in any of the three channels, so the theoretical Laplacian $D(\boldsymbol{x})$ should be calculated as the sum of absolute values of theses three Laplacians:

$$D(\boldsymbol{x}) = \|D(\boldsymbol{x}^R)\| + \|D(\boldsymbol{x}^G)\| + \|D(\boldsymbol{x}^B)\| \tag{12}$$

Li, Shaohua *et al.*[11] proposed to approximate the theoretical result using simple summation of the three Laplacians:

$$D(\boldsymbol{x}) = D(\boldsymbol{x}^R) + D(\boldsymbol{x}^G) + D(\boldsymbol{x}^B) \tag{13}$$

Using this approximation, Li, Shaohua *et al.* claimed that they have not observed any quality degradation of the synthetic image.

Smoothing with a pooling layer The Laplacian filter is very sensitive to noise so smoothing the image with an average pooling layer before the Laplacian filter can better preserve the actual detailed structure of the content image. Besides, we can save the memory to $1/p^2$ by applying a $p \times p$ pooling layer overhead of the Laplacian loss. The size p is also a parameter that can be fine-tuned in order to get a better synthetic image.

Combining Multiple Laplacians With a kernel in bigger size, the pooling layer can squeeze a bigger area into a pixel. With a Laplacian filter overhead of the such pooling layer can preserve the structures in a wider region. Hence, it is a great idea to combine multiple Laplacians with different sizes of the Laplacian filter into the Laplacian loss, which may capture the structures in different scales. The Laplacian loss can be revised as:

$$L_{total}(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{x}) = \alpha L_{content}(\boldsymbol{p}, \boldsymbol{x}) + \beta L_{style}(\boldsymbol{a}, \boldsymbol{x}) + \sum_k \gamma_k L_{lap,k}(\boldsymbol{p}, \boldsymbol{x}) \tag{14}$$

where $L_{lap,k}$ is the Laplacians overhead of a pooling layer with size $p_k \times p_k$ and γ_k is the corresponding wight.

4 Results

4.1 Tuning the Content/Style Ratio and Picking Layers

The content image we used in this part is the photography of Stockholm, while the style image is the Chinese painting named "Dwelling in the Fuchun Mountains" by Gongwang Huang, as is shown in Fig.3 and Fig.4.

As is shown in Equation 5, there are two parameters α and β in the loss function controlling the strength of the content image and style image. Their ratio α/β reflects the trade-off between the content and style information. When the content/style ratio α/β is larger, the output image remain more information from the content image, but with a less noticeable style. However, when the content/style ratio α/β is smaller, the output information will contain more style information, but less content information.

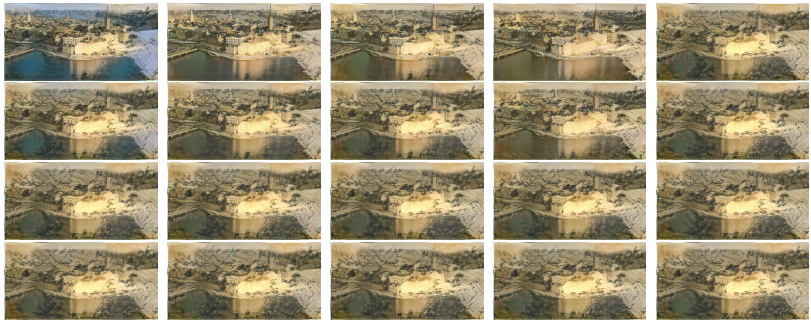
Moreover, picking different layers of the VGG-19 network will also has an influence on the output image performance. In the VGG network, the deeper layer represents more high-level information of the image, which indicates losing

Fig. 3. Content Image

Fig. 4. Style Image

more low-level detailed information. As we can notice in Fig.5, the detailed structures, for example the edges of the buildings, can be preserved less and less as the picking layer for content image goes deep.

The output images with different content/style ratio α/β and different picked layers can be shown in Fig.5.



(a) conv1-2 (b) conv2-2 (c) conv3-2 (d) conv4-2 (e) conv5-2

Fig. 5. Reconstruction capability of different contentstyle ratios α/β and representation at different layers, the α/β from top to bottom is $10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}$ respectively

4.2 Multi-style Transfer

In this part, we try out blending between the Kandinsky and the Starry Night, as well as blending between the Seated Nude and the Scream, into our chosen content image Tubingen. As we can see from the Fig.6, the generated images contains both styles of the above style images. Depending on the loss weights, the two styles in the same image can have different contributions. An interesting observation is that, each style tends to occupy a certain part in the image.

(a) Style1 (b) 80% / 20% (c) 50% / 50% (d) 20% / 80% (e) Style2

Fig. 6. Results of Multi-style transfer. By setting the weights of style losses, we can control the effects that each style image has on the synthetic image.

4.3 Color Preserving Style Transfer

In this part we choose the Tubingen as the content image and the Starry Night as the style image.

The luminance-only method for color preserving style transfer works very well according to the result of our experiments. Fig.7 is the comparison of the ordinary neural style transfer and color preserving style transfer with the same content image and style image.

We can easily find out that after the ordinary neural style transfer, the color of output image is dark blue, because of the influence of the style image. However, color preserving style transfer can keep the output image to have similar color with the content image.

4.4 Total Variation Regularization

In Fig.8 we compare the performance of the ordinary neural style transfer and the neural style transfer with total variation regularization. The total variation regularization works as a denoising tool, which can smooth the output image. However, in our cases the noise in content image do not have much influence on the quality of final output. Thus, it shows very little difference in Fig.8 between the output with and without total variation regularization. The distortion problem in output image will be better solved by the Laplacian-Steered Neural Style Transfer method discussed below.

4.5 Laplacian-Steered Neural Style Transfer

From Fig.9 we can note from the comparison between the total loss without Laplacian term, e.g. 9(a) and Laplacian loss added in the total loss function, e.g. 9(e), some basic shapes in the building of the content image are preserved in the generated image. Besides, We can see some slight differences with the different Laplacian loss weights γ from Fig.9(b)-(e), nevertheless reducing γ will not change the generated image in a significant way[2].

Fig. 7. Color-preserving Style Transfer: The top-left image is the content image; the top-right image is the style image; the bottom-left image is the ordinary neural style transfer output; and the bottom-right image is the color preserving transfer output.

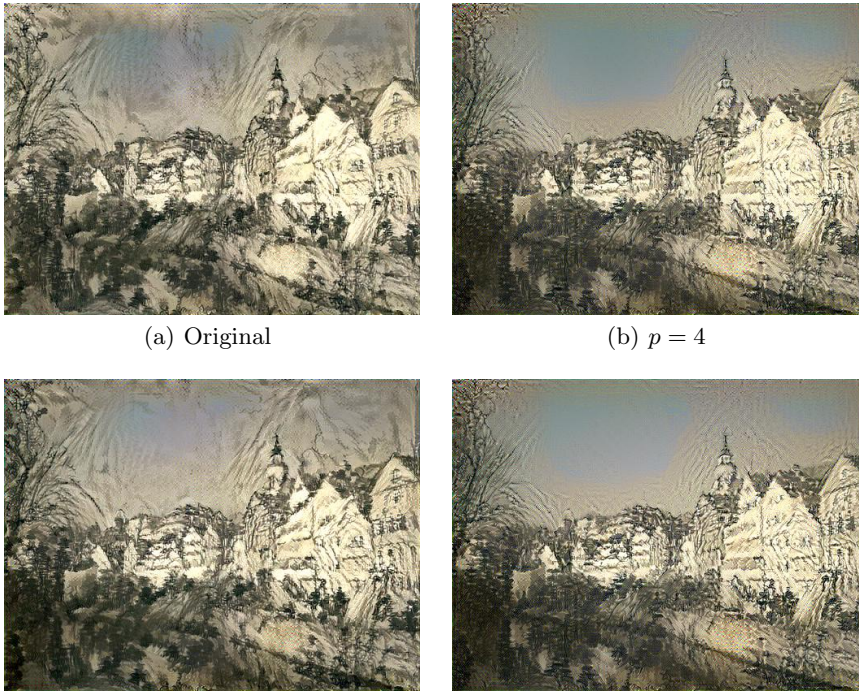
(a) Without total variation (b) Total variation

Fig. 8. Comparison of neural style transfer with and without total variation regularization

As we mentioned in the method part, the size of pooling layer p have an influence on the quality of the image. A larger size filter will capture the low-level structures in a larger scale, thus multiple Laplacians might have a better result. As we can see in the Fig.10, the generated image on the basis of $p = 4$ Fig.10(b) preserve most of the local structures, while Fig.10(c) did not succeed in this due to the size of filter too large. The combination Fig.10(d) not only preserve the detailed structures of the content image, but also achieve the best stylization.

(a) Without Laplace (b) γ=0.1 (c) γ=1 (d) γ=10 (e) γ=150

Fig. 9. Laplace-Steered Neural Style Transfer with different parameter γ setting



(a) Original (b) $p = 4$

(c) $p = 16$ (d) $p_1 = 4, p_2 = 16$

Fig. 10. Comparison of Laplacian with difference pooling sizes

5 Conclusions

In this project, we first replicate Gatys *et al.*'s work to realize the neural style transfer system. We find out that by changing the content/style ratio in the loss function, as well as using different layers of the VGG-19 network, we can get different output image.

Secondly, we realize some further functions based on the original theory of neural style transfer. The multi-style transfer can synthesis an image with content from one image, and style from two or more other images. By changing the weights before the loss of the style images, we can get the output we like. The

color-preserving transfer can help the output image get rid of the influence of style image on its color. By using the method of luminance-only transfer, we realize this function ideally.

Moreover, we improve the loss function by adding a total variation regularization term onto it. This term can denoise the content image, and make the output image more smooth. However, the output image does not change much after this altering.

Finally, we add a Laplacian term to the loss function to get rid of some distortions and irregular artifacts in the output image. Our results show that this method can help maintain the pixel level information while remaining the transferred style. Also, according to the experiments we suppose that transfer results may depend on the initial content image or style image somehow.

References

1. Gatys, L., Ecker, A., Bethge, M.: A neural algorithm of artistic style. Journal of Vision **16**(12) (August 2016)
2. Li, C., Wand, M.: Combining markov random fields and convolutional neural networks for image synthesis. (January 2016)
3. Johson, J., Alahi, A., Fei Fei, L.: Perceptual losses for real-time style transfer and super-resolution (2016)
4. Ulyanov, D., Lebedev, V., Vedaldi, A., Lempitsky, V.: Texture networks: Feed-forward synthesis of textures and stylized images. (March 2016)
5. Jing, Y., Yang, Y., Feng, Z., Ye, J., Yu, Y., Song, M.: Neural style transfer: A review. arXiv preprint arXiv:1705.04058 (2017)
6. Zhang, H., Dana, K.: Multi-style generative network for real-time transfer. (March 2017)
7. Dumoulin, V., Shlens, J., Kudlur, M.: A learned representation for artistic style. (October 2016)
8. Chen, T.Q., Schmidt, M.: Fast patch-based style transfer of arbitrary style. (December 2016)
9. Ghiasi, G., Lee, H., Kudlur, M., Dumoulin, V., Shlens, J.: Exploring the structure of a real-time, arbitrary neural artistic stylization network. (May 2017)
10. Gatys, L.A., Bethge, M., Hertzmann, A., Shechtman, E.: Preserving color in neural artistic style transfer. arXiv preprint arXiv:1606.05897 (2016)
11. Li, S., Xu, X., Nie, L., Chua, T.S.: Laplacian-steered neural style transfer. (July 2017)