STYLE TRANSFER TRILOGY: FROM FIXED TO VARIABLE

MACHINE LEARNING FINAL PROJECT PAPER

Xinyue Liu
NYU Shanghai
xl2151@nyu.edu

Yuejiao Qiu
NYU Shanghai
yq639@nyu.edu

May, 2021

ABSTRACT

The development of Convolutional Neural Networks (CNN) has enabled the computer to extract abstract features of images, and therefore leads to the emergence of filter apps, such as Prisma, which can blend an aesthetic style with a content picture. This study aims to implement three different algorithms to realize the artistic style transformation. We will first use VGG19 for a fixed pairwise style and content in an iteration approach. Then we adopt image transfer network (TransformNet) to train a model that is able to transfer a fixed style to variable content images within a second. For further exploration, we find that Meta network can output the weight which can be directly used for any style and any content.

Keywords Neural style transfer · VGG19 · Image transform network · Meta Network

1 Introduction

Since AlexNet was introduced in 2012, the utilization of Deep Convolutional Neural Networks has skyrocketed in computer vision. Convolutional Neural Networks is able to extract abstract features of images; therefore, it empowers computers to perform some complex tasks, for example, image style transfer. Generally, image style transfer aims to synthesising a texture from a source image while constraining the texture synthesis in order to preserve the content of a target image. However, compared with traditional machine learning problems, this task is formidable because it is hard to describe the style and content mathematically. And it is also challenging to evaluate a synthesized image due to the indefinite representation of images.

Gatys et al. (2016) suggest that Convolutional Neural Networks is powerful to extract high-level semantic information from natural images, which is a major breakthrough in image synthesis and manipulations. And this also trigger the development of neural style transfer (NST).

There are many approaches to address this problem, but they roughly belong categories: the image-optimization-based (IOB) online neural methods and model-optimization-based (MOB) offline neural methods. The basic idea of the IOB-NST algorithms is to recombine the extracted style and content information and then reconstruct a stylized result iteratively. IOB-NST algorithms are computationally expensive due to the iterative optimization method. The most representative algorithm is Neural Algorithm for Artistic Styles proposed by Gatys et al. The second method MOB-NST algorithms address the speed and computational cost through a fee-forward network optimized over a large set of content images for one or more style images. The first two MOB-NST algorithms are came up with by Johnson et al. and Ulyanov et al. respectively, which they share a similar idea to pre-train a feed-forward style-specific network and produce a stylized result with a single forward pass at testing stage.

For model implementation, we first adopt IOB-NST algorithms proposed by Gatys et. al (2016) to realize style render of a fixed style to a fixed content. Then to preserve better visual quality as well as runtime efficiency, we follow Johnson et. al (2016) and Ulyanov et. al (2017), using inmage transform network and instance normalization for real-time style transfer. Besides, we also find that Shen et al. (2018) propose a noval method to generate specified network parameters through one feed-forward propagation in the meta netwroks for NST. However, due to the large storage space for training, we do not complete the implementation of NST via meta networks.

2 Data

The first model, basic style transfer using IOB-NST algorithms, inputs one specific style image and one content image into the iteration loop. Therefore, in the first model, we can use an arbitrary pair of style and content images for each transfer. However, for the second generative model, since we are going to train the model for real-time style transfer, it is of great importance to use a sufficiently large dataset as content images to train the model. Therefore, we choose to use LabelMe, which is a large dataset created by the MIT Computer Science and Artificial Intelligence Laboratory containing 187,240 images. As for style images, we use famous paintings as styles, which are available on Kaggle (`https://www.kaggle.com/momincks/paintings-for-artistic-style-transfer`).

3 Models and algorithms

3.1 Image representations

By reconstructing representations from intermediate layers of the VGG-19 network, Gatys et al. (2016) observe that a deep convolutional neural network is capable of extracting image content from an arbitrary photograph and some appearance information from the well-known artwork. Based on this observation, we can build the content component of the newly stylised image by penalizing the difference of high-level representations derived from content and stylised images, and further build the style component by matching Gram-based summary statistics of style and stylised images, which is derived from their proposed texture modelling technique. Both the content loss and style loss are determined from the activations of atrained convolutional neural network. The loss at a single layer is the Euclidean ($\mathcal{L}2$) distance between the activations of the content image and the activations of the output image. Therefore, a loss function can be defined as follows.

Let $\vec{p}$ and $\vec{x}$ be the original image and the generated image, and P^l and F^l be their respective feature representation in layer l. The content loss is defined as follows:

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

To obtain a representation of the style of an input image, we use a feature space designed to capture texture information. These feature correlations are given by the Gram matrix $G^l \in^{N_l \times N_l}$, where G_{ij}^l is the inner product between the vectorized feature maps i and j in layer l:

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l \tag{2}$$

By including the feature correlations of multiple layers, we obtain a stationary, multi-scale representation of the input image, which captures its texture information but not the global arrangement.

Let $\vec{a}$ and $\vec{x}$ be the original image and the generated image, and A^l and G^l be their respective style representation in layer l. The contribution of layer l to the total loss is then

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

where he constant $\frac{1}{4N_l^2 M_l^2}$ is a normalization term.

Hence, the total style loss is the weighted sum of the style loss at each layer.

$$L_{style}(\vec{a}, \vec{x}) = \sum_l w_l \cdot E_l \tag{4}$$

Therefore, the objective function is the weighted sum of style and content loss:

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

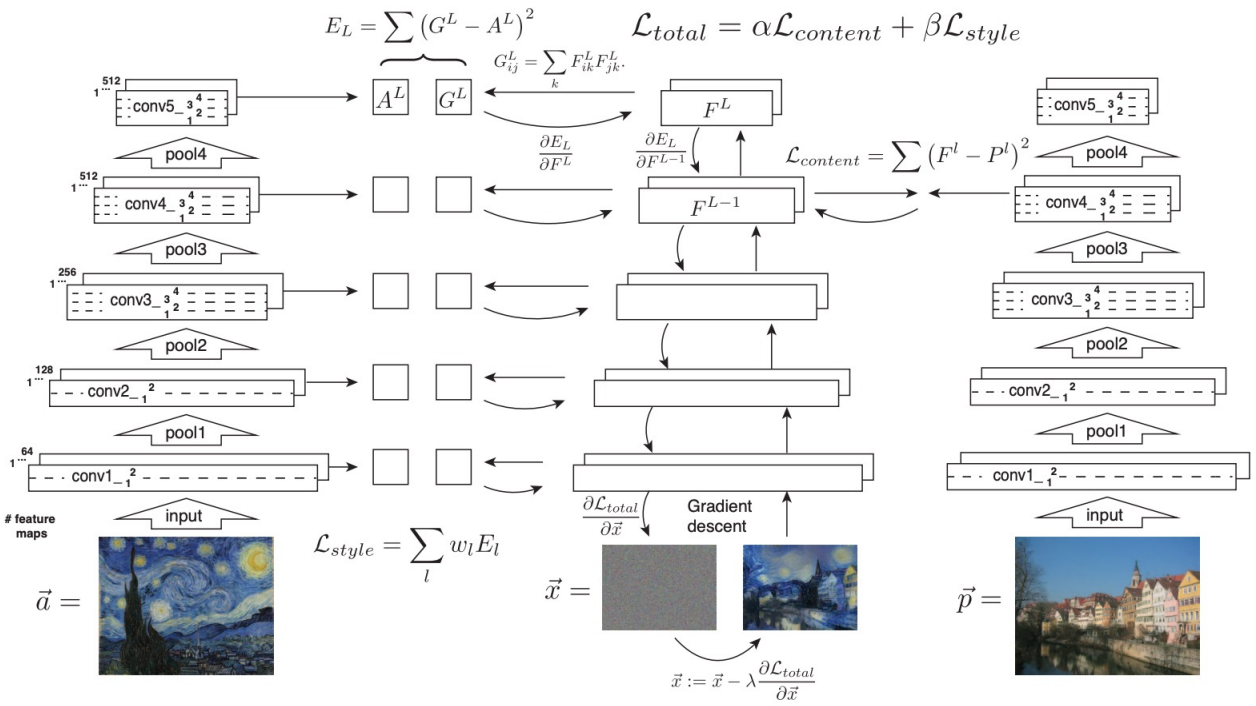

Figure 1: Style transfer algorithm

3.2 Basic style transfer

Basic style transfer ultilizes a pre-trained VGG19 as the loss network, which includes 16 convolutional and 5 pooling layers. We normalize the network by scaling the weights so that the mean activation of each filter over images and positions is equal to one. And we replace max pooling by average pooling. According to Gatys et al., average pooling yields slightly more appealing results. We also remove the fully connected layer, since we can directly output an image of the same dimensions.

The choice of style and content layers is a very important factor in style transfer because different positions and numbers of layers can bring about very different results. According to Gatys et al., we will use `{relu1_1, relu2_1, relu3_1, relu4_1, relu5_1}` as style layers and `{relu4_2}` as the content layer. And we use features in these layers to compute loss. In order to transfer the style of an artwork $\vec{a}$ onto a photograph $\vec{p}$, we synthesise a new image that simultaneously matches the content representation of $\vec{p}$ and the style representation of $\vec{a}$, (Fig 1). Thus we jointly minimize the distance of the feature representations of a white noise image from the content representation of the photograph in one layer and the style representation of the painting defined on a number of layers of the Convolutional Neural Network. Note this algorithm does not require ground truth data for training; thus we adopt LBFG_S as the optimizer to update the gradient iteratively to find the stylized image that minimize L_{total}.

3.3 Fast style transfer via Transform Net

Although the previously-discussed IOB-NST is able to yield pretty impressive results, the efficiency is a big issue because we need to input a new image and go over the iterations every time if we want to use either a new style or content image. Therefore, we build a model to perform real-time style transfer by using MOB approaches. The first two MOB-NST is proposed by Ulyanov et al. and Johnson et al. (2016), both of which pre-tain a feed-forward style-specific network and produce a stylized result with a single forward pass at testing stage. The only difference is network architecture that Johnson et al. follows the network proposed by Radford et al. (2015), whereas Ulyanov et al. use a multi-scale generator network. For simplity, we choose to implement fast style transfer by using Transform Net proposed by Johnson et al.

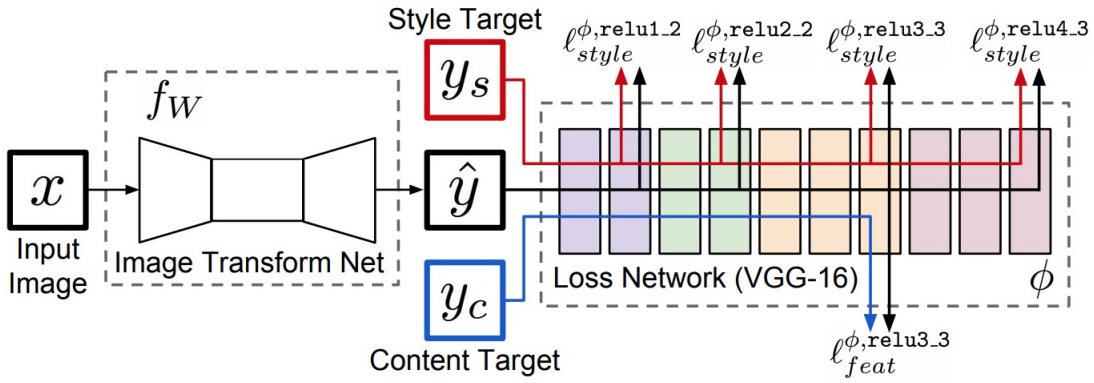

Figure 2: System overview. We train an image transformation network to transform input images into output images. We use a loss network pre-trained to define perceptual loss functions that measure perceptual differences in content and style between images. The loss network remains fixed during the training process. Note the original model uses VGG16 as loss network, but we actually exploit VGG19.

The structure of image transform network is as follows. The network inputs an image tensor of shape $H \times W \times 3$. First, the image tensor is passed to three conv blocks. With channel increasing to 128, they transform the input image into higher dimensional space and downsample the feature maps through stride-2 conv layers. Following the three conv layers, there is a sequence of five residual blocks. He et al. (2015) argue that residual connections make it easy for the network to learn the identify function; therefore, the five residual blocks are able learn to add features on the original image for style transformation. Note that all the five residual blocks use 128 filters and thus stays within the $H \times W \times 128$ dimensional space. Finally, there are two conv transpose layers with stride 1/2 for upsampling, which also brings the feature map into lower-dimensional space. Then a conv layer with tanh as activation turns the output to RGB image.

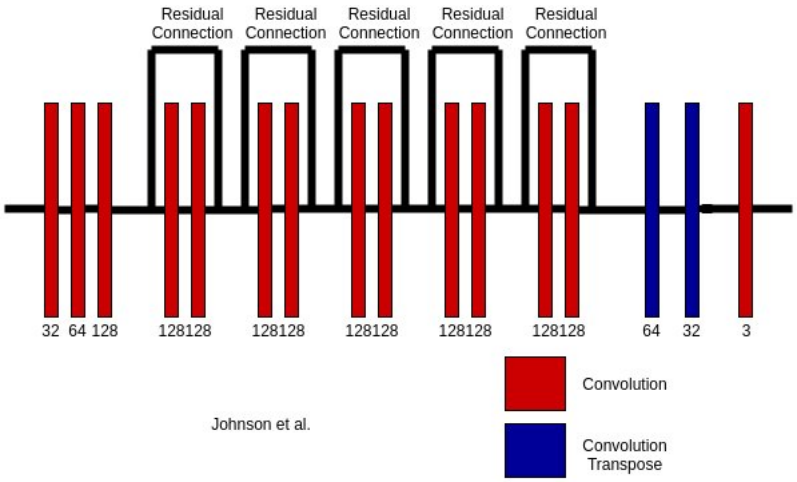

Figure 3: Transform Net architecture. The first convolutional layer has 32 filters of size 9×9, the second has 64 filters of size 3×3 and the third has 128 filters of size 3×3. The residual blocks contain convolutional layers having 128 filters of size 3×3. The first transpose convolution layer has 64 filters of size 3×3 and the second has 32 filters of size 3×3. The last convolutional layer has 3 filters of size 9×9. "ReLU" non-linearity has been applied to all the layers except the last convolutional layer.

There are also some other implementation details to be noted. We replace batch normalization by instance normalization proposed by Ulyanov et al (2017). Instance normalization is equivalent to batch normalization when the batch size is one. It is shown that style transfer network with instance normalization layer converges faster and produces visually better results compared with the network with batch normalization layer. Ulyanov et al (2017). believe that the superior

performance of instance normalization results from the fact that instance normalization enables the network to discard contrast information in content images and therefore makes learning simpler.

3.4 Meta networks for neural style transfer

To find an image transformation network $\mathcal{N}(;w)$, which is parameterized by w for a given style image I_s, it needs tens of thousands of SGD iterations to get a satisfied network. That is to say, there is always an image transformation network N for every style image I_s. Luan et al. (2017) propose that, it is straightforward to train an end-to-end neural networks by minimizing a pre-defined loss function. The key idea is to get the transformation network $\mathcal{N}(;w_*)$ by a meta network:

$$Meta\mathcal{N} : I_s \rightarrow \mathcal{N}(;w).$$

Here the meta network is parameterized by θ. In the training stage, the parameter is optimized by minimizing the empirical loss across a dataset of content images $\mathcal{D}_c$ and a dataset of style images $\mathcal{D}_s$ by

$$\min_{\theta} \sum_{I_c \in \mathcal{D}_c} \sum_{I_d \in \mathcal{D}_d} L_{total}$$

4 Experiments

4.1 Basic style transfer

In our experiment, we use Python Pytorch 0.4.0 to implement basic style transfer. Since basic style transfer relies on IOB-NST and does not require training, we choose some arbitrary styles and content images. Note that there are also several hyperparameters included in the model. We simply follow Gatys et al. that we assign content loss weight $\alpha = 1$ and style loss weight $\beta = 1 \times 10^6$. The syntheses are outlined below:

Figure 4: Output of basic style transfer

Many scholars have mentioned that there is a lack of systematic and quantitative way to evaluate the output of image style transfer problem. And many of them give an intuitive evaluation of their models built. Therefore, in our model, the evaluation is mainly based on perception and intuition. We can see that the basic style transfer can roughly blend the style and the content, as it successfully transfer color and texture of the style image, while preserving the content.

However, one of the biggest drawbacks of this model is that, each time we want to use either a new style image or a new content image, we need to input the new image into the model and use gradient descent to minimize the loss iteratively, which is not efficient. In our experiment, we manually set the number of iterations equal to 800, and the average time to generate the output image is 65.03s, even with a GPU. In conclusion, the IOB basic style transfer is a both style-specific and content-specific model, which is not able to realize real-time style transfer. It might be optimal for limited styles and contents. Thus, we build fast style transfer which will be talked in the next section.

4.2 Fast style transfer via Transform Net

4.2.1 Training

In this experiment, Image Transform Network is implemented in Python Torch 0.4.0. We also adopted GPU supported by Google Colab. In the training process, since the LabelMe dataset is very large with 187,240 images, we choose to use only a batch of it to train the model, which contains roughly 25,000 images. In terms of the hyperparameters, the content loss weight is still 1. However, we adjust style loss weight to be 1×10^5 for more ideal outputs, which different from the 5 given by the official open source. This might be due to the discrepancies in the data preprocessing stage. The weight for total variation loss is 1×10^{-6}. Moreover, in the optimizer, We use Adam with a learning rate of 1×10^{-3}.

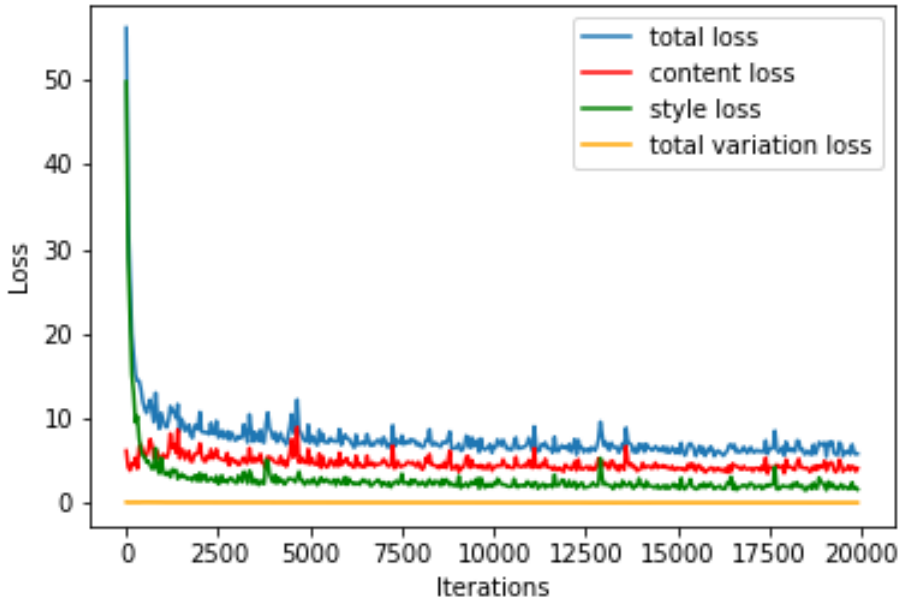

Figure 5: Loss during training

The loss during training is plotted in Figure 5 above. On the one hand, we can see that content loss does not change significantly because of the initialization. On the other hand, style loss start to drop greatly after about 1000 iterations, and then the decrease is not very significant. We also find that more iterations generally lead to more appealing; the blend of the style and content is better. As a result, we still add more iterations, even though the loss decreases at a very low speed.

4.2.2 Outputs

In line with the basic style transfer model, there is still a lack of quantitative evaluations. The outputs of the fast style transfer model are presented in Figure 6. In order to fully test the model performance, we do not limit the test content images to be scenery, but also choose a few pictures with figures. As it can be seen in the output grid, we can assert that the model is performing well because in terms of the style, it not only transfer the color but also the texture of the style image, while in terms of the content, the objects in the content image are well preserved. Even using the Mosaic style which is shown in the last row, we can still observe a clear shape of the buildings and the figures. As for the runtime, as this is a MOB-NST that we build a fee-forward network that only requires one pass in the testing stage, the average time to apply a specific style to a random content is about 0.62s, which partially realize real-time style transfer.

Figure 6: Output of fast style transfer

5 Conclusion

In this project, we have tried to implement two different style transfer models sequentially using different approaches and algorithms and further explore the style transfer algorithm (Meta Network). In order to produce better results, we modified the networks along implementation by using instance normalization and tahn activation. Our final model with Image Transform Network, is able to blend a certain style into an arbitrary content image within a second, which largely increase efficiency.

Nevertheless, although fast style transfer is able to produce satisfying results within a very short time, we still need to redo the training process, if we want to transfer a new style, which is very time-consuming. To address this issue, we do find a solution for real-time style transfer regardless the style, which is Meta Network. The key idea is to train a network to minimize the empirical loss using large datasets of both style images and style images. But due to time and storage limits, we do not complete the implementation. However, this finding is still remarkable.

Even though it is feasible to perform real-time transfer of any content and any style, each time we can only synthesize one style and one content. Then it is natural to consider blend multiple styles into one content image. The answer lies in the weight of differnt styles. Simply by adjusting the weights of different styles, we are able to blend the styles proportionally into the content image.

References

[1] Ulyanov, D., Lebedev V., Vedaldi, A., and Lem-pitsky, V., "Texture Networks: Feed-forward Synthesis of Textures and Stylized Images", *arXiv e-prints*, p.arXiv:1603.03417, 2016.

[2] Gatys,L. A., Ecker,A. S., and Bethge, M., "Image style transfer using convolutional neural networks", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[3] Jing,Y., Yang, Y., Feng,Z., Ye, J., Yu, Y., and Song, M., "Neural style transfer: A review", *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2019.

[4] Shen, F., Yan, S., Zeng, G., "Neural Style Transfer via Meta Networks"