A Neural Algorithm of Chinese Calligraphy Style

Yue Peng, Ludan Zhang
School of Public Health

Introduction

Image Style Transfer has been a heated debate these two years since this paper was published. Usually, it is too unclear to transfer a style to another in Machine Learning. It is interesting that we could utilizing deep neural network models to define the style mathematically and transfer it to any image. Leon's[1] paper studied a method to generate new images using pre-trained Convolutional Neural Network(VGG-19) based on style and content images. The feature maps (differently filtered versions of the input image) generated by Convolutional Neural Network which is well known as a network structure processing visual information can represent content and style extracted from two different images. Chinese calligraphies are well known as having many styles. Thus, we would like to find a neural algorithm specifically for Chinese calligraphy to enable the application of calligraphy styles. To do so, a special Convolutional Neural Network needs to be trained before applied in our neural algorithm. After that, we can use this pre-trained model to finish our style transfer on Chinese character.

Method

Train Convolutional Neural Network

The data were from are images of 3755 Chinese characters from the website www.nlpr.ia.ac.cn/databases/handwriting/Home.html . For each character, there were about 240 images for training and 60 images for test. Our mission was to develop a neural network to recognize the Chinese characters. To control the training time, we chose only 200 characters to do the task. Our network was designed as following:

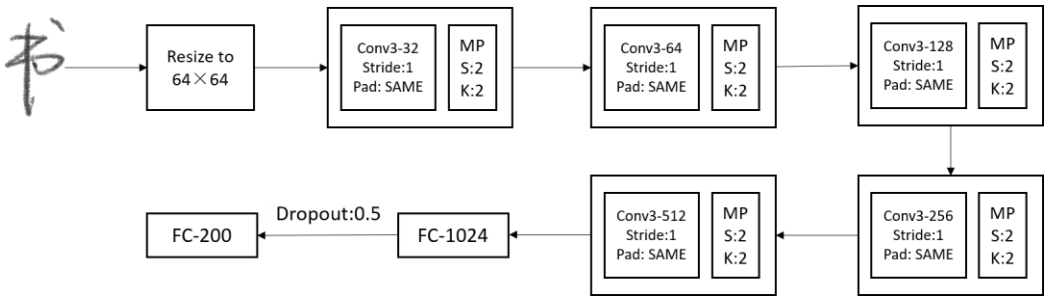

Figure 1. Pre-trained Neural Network. All conv and fc layers were followed by a ReLU activation layer.

After training for 20 epochs with Adam Optimizer with learning rate of 0.001, the network achieved 90% accuracy of recognizing the characters. And this network was saved to be used in our neural algorithm in the future.

The Neural Algorithm

After training the neural network, we can feed the style and content images into this neural network to get their features. We wanted to generate an output image which can keep the content

of the content image and have the style of the style image. Figure 2 shows the features we got after feeding the images through different layers of the pre-trained network.

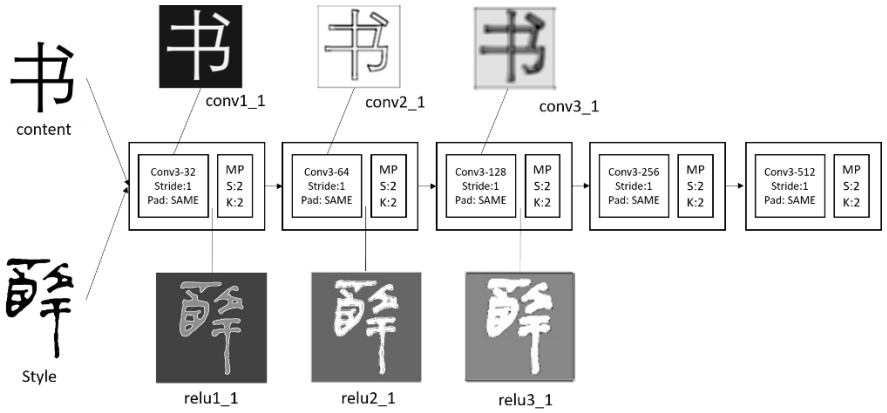

Figure 2. The features after feeding the images through different layers

We can choose the layers through which the style and content images to be fed through. Different layers can result in different style and content features to be captured, thus can result in different output images. We choose the $conv1\_1$ and $conv1\_2$ for content images and $relu1\_1$, $relu2\_1$, $relu3\_1$ for style images as shown in the Figure 2 in our algorithm. The layers were chosen through attempts and experience. The lower level layers can keep fine structures of the original images. To train the output images, we needed to minimize the loss defined as:

$$loss = content\ weight \times content\ loss + style\ weight \times style\ loss + variation\ weight \times variation\ loss$$

Where three kinds of losses needed to be computed. First, the output image was initialized using $tf.truncated\_normal(stddev = 0.1)$ and the same shape as content image.

a) content loss

Define the layers the content images fed through as *content layers*, and the features generated by these layers as *content features*. The initialized output image needed to go through the content layers as well, and the generated features were defined as *image content feature*. The content loss can be expressed as:

$$content\ loss = \sum_{i\ in\ content\ layers} 2 \times \frac{|content\ feature_i - image\ cotent\ feature_i|^2}{ize\ of\ content\ \ feature}$$

b) style loss

Define the layers the style images fed through as *style layers*, and the features generated by these layers as *style features*. The initialized output image needed to go through the style layers as well, and the generated features were defined as *image style feature*. Since the output image and the style image had different sizes, the *style feature* and *image style feature* had different sizes in the first, second and third dimensions for a certain layer but had the same size in the fourth dimension. Thus, the *style feature* and *image style feature* were all transformed into 2 dimension matrixes. For example, if style feature for a certain layer has the size of $[k, h, w, d]$, the *transformed style feature* would have the size of $[k \times h \times w, d]$, then the matrix will be

multiplied by its transpose to arrive at a *transformed style feature* of size $[d, d]$ finally. It is the same for *image style feature*. The style loss can be expressed with the *transformed style feature* and *transformed image style* feature as:

$$style\ loss = \sum_{i\ in\ style\ layers} \frac{|transformed\ style\ feature_i - transformed\ image\ style\ feature_i|^2}{size\ of\ style\ feature\ \times size\ of\ image\ style\ feature}$$

c) variation loss

Assume the output image has the size of $[h, w]$, then the variation loss can be defined as:

$$variation\ loss = \frac{|image[1:h, 0:w] - image[0:h-1, 0:w]|^2}{h} + \frac{|image[0:h, 1:w] - image[0:h, 0:w-1]|^2}{w}$$

In all, the structure of the neural algorithm can be summarized in Figure 3.

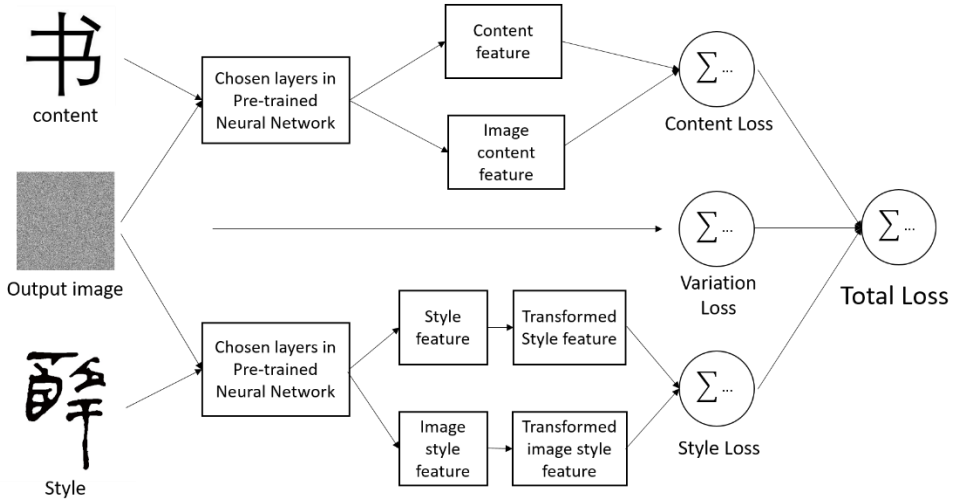

Figure 3. The structure of the neural algorithm

Train the Output Image

The weights used in the loss function were set as *content weight = 5, style weight = 5e2, variation weight = 10.* To train the output image, we used the Adam Optimizer(learning rate:0.01, beta1:0.999, beta2:0.9999) to minimize the loss with 1000 iterations. The output image was saved finally.

Results

Train Convolutional Neural Network

We ran experiments on a 200-class using convolutional networks with different depth and filter numbers. After making the network deeper with a quite small filter size and quite large filter numbers, we got a 90% accuracy running on Amazon EC2 instance with NVIDIA Tesla K80.

The Neural Algorithm

We picked two characters 书法(means: calligraphy) to do our experiments. According to the Figure 4, we could see that the results came from original neural algorithm[1] was not able to

capture the calligraphy style and apply it to the content image correctly. We would like to keep the content as much as possible by just changing the strokes style according to the style image. So, we changed the VGG-19 used in the original neural algorithm[1] to a pre-trained convolutional network focusing on Chinese characters to get a pleasant result.

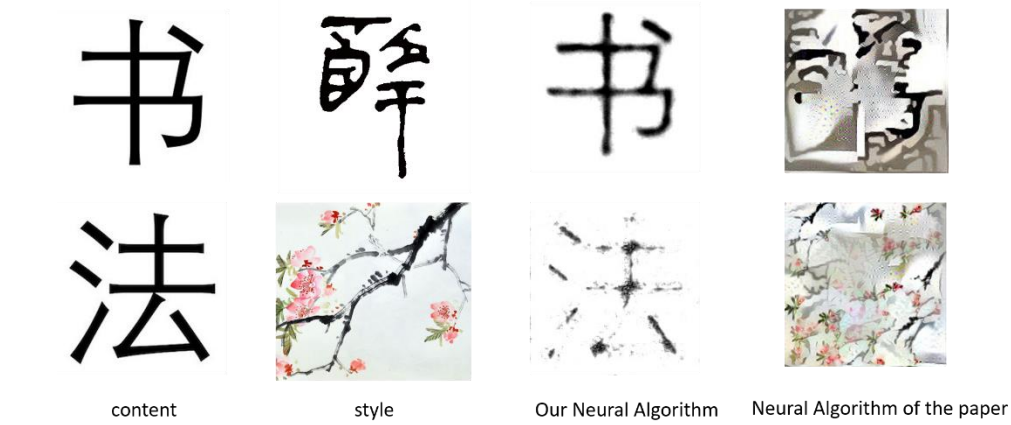

Figure 4. The results of our neural algorithm. The new 书 got the style of the strokes in the style images and the new 法 got the style of the twig in the style image.

Discussion

The pre-trained process of our convolutional network was limited since we just chose 200 out of 3755 characters to train our network. The generalization capacity of our algorithm would be better if we could train more kinds of characters.

The output image can be varied a lot with different size of content and style images. When the style image is 5 times size of the content image, we can get the optimal output image. This may because of limitation of pre-trained network. Also, changing the weights used in the loss function can help the algorithm adjust to different image sizes.

Reference

[1] Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "A neural algorithm of artistic style." arXiv preprint arXiv:1508.06576 (2015).