A Neural Algorithm of Artistic Style

E4040.2016Fall.PPAP.report
Zhuo Kong zk2202, XunXue xx2241, Yuanxu Zhu
Columbia University

Abstract

In this project we implement a neural algorithm with Theano to explore artistic style transfer by performing style and content representation respectively and then combine them together, during the process of which we discuss the results with different parameter settings as well as optimization approach.

1. Introduction

Nowadays, Deep neural network has been widely implemented in many areas of visual perception, especially convolutional neural network. VGG-19 network is a deep neural network widely used in visual perception areas. With the imagenet pretrained VGG network, we can perform tasks on extracting features of content or style from different convolutional layers, which leads to the methodology of reconstructing an image combining the content and style feature. In this article, we will introduce this methodology and the implementation, and discuss the relevant results.

2. Summary of the Original Paper

2.1 Methodology of the Original Paper

In the original paper, the author proposes that each layer in the network defines a non-linear filter bank, and to visualize the information that is encoded in the layer's hierarchy we can perform gradient descent on a white noise image to find another image that matches the feature responses.

For style representation, the author use Gram matrix to measure it as suggested by the original paper. In order to find another image that matches the style representation of the original image by minimising the mean-squared distance between the entries of the Gram matrix from the original image and the Gram matrix of the image to be generated.

To generate the images that mix the content of a photograph with the style of a painting, the author jointly minimise the distance of a white noise image from the content representation of the photograph in one layer of the network and the style representation of the painting in a number of layers of the CNN, which means that they combines the loss function of style and contend loss in their implementation of style transfer.

2.2 Key Results of the Original Paper

1. Representations of content and style of image in the convolutional neural network are separable and image created by this network has the content of the original photo and style of artwork.

Fig 1 Result of *A Neural Algorithm of Artistic Style*

2. According to theories and tests, reconstruction of photos from lower layers is almost perfect while photos in higher layers loss detailed pixel information. And in CNN, style representations need to be computed correlations between different features in different layers.

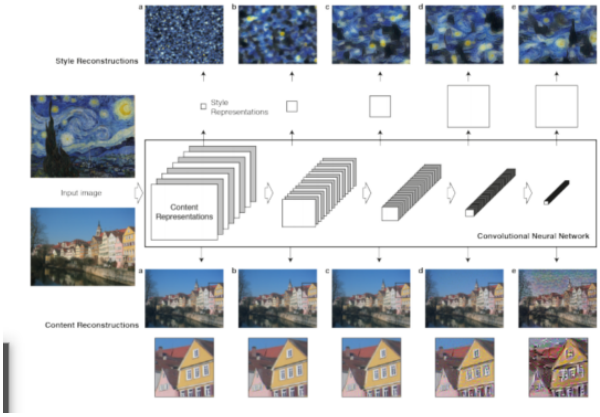


Fig 2 Result of *A Neural Algorithm of Artistic Style*

3. They find out that the local image structures captured by the style representation increase in size and complexity when including style features from higher layers of the network. And results varied in choosing the ratio of the emphasis of content and art of the image.

Fig 3 Result of *A Neural Algorithm of Artistic Style*

3. Methodology

In this project, we build the VGG network architecture using theano, then we reconstruct the image from the feature map in each layer in the network, based on which we discuss the results of content reconstruction and style reconstruction. Then we perform the style transfer task using the methodology from paper and discuss the results when putting different emphasis on style and content.

3.1. Objectives and Technical Challenges

3.1.1 Objectives

The objectives of this project is rebuild the architecture of the original paper and get similar results, which include the result of content reconstruction and style reconstruction for different layers, and different results of style transfer when modifying the parameter to emphasis differently on content and style.

3.1.2 Technical Challenges

For the architecture building part, we should use theano to build vgg19 neural network, the main issue is how to load weight to our architecture. Since we don't have experience of loading weight before, it takes us a lot of time to figure it out.

For the loss function part, it is pretty hard to implement the formula in the original paper to theano code. It also took us very long time to figure out why there are so much noises on the image when performing training without the total variation loss.

For the optimization algorithm part, since we don't know bfgs before, we spend many time experimenting with adam, which is much more slower. The scipy function for bfgs is also very confusing and hard to implement.

3.2. Problem Formulation and Design

3.2.1 Build network architecture

The first step is building VGG architecture. we use theano.tensor.nnet.conv2d as convolutional layer and theano.tensor.singal.pool.pool_2d as pooling layer to get a VGG-Network with 16 convolution layers and 5 pooling layers. And we do not use fully connected layers. Then we get weight from pre-trained weight from lasange online source and load them by passing the parameter into build_net function as array. We replace the max-pooling operation by average pooling operation to improve the gradient flow as suggested by the original paper.

3.2.2 Image preprocessing

After that we perform the image preprocessing. This part is not mentioned in original paper but is necessary when performing tasks relevant to image. Firstly resize the image to make the width and height of the image at least as large as a constant(we set it as 600 in this project), then we cut the redundant part of the image to make it exactly 600*600 size. Then we swap and add axes to the image and make it finally the size of (1,3,600,600) to make fit the input size of CNN.

3.2.3 Define loss function

We define the loss function with the formula from the original paper. In the original paper, the author proposes that each layer in the network defines a non-linear filter bank, and to visualize the information that is encoded in the layer's hierarchy we can perform gradient descent on a white noise image to find another image that matches the feature responses. So we define the squared-error loss between two feature representations which has already be proposed by "Summary of the Original Paper" part.

The loss function is defined in the paper as below.

Content squared-error loss:

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

Derivative of this loss to the activations in layer l:

$$\frac{\partial\mathcal{L}_{content}}{\partial F_{ij}^{l}}=\begin{cases}\left(F^{l}-P^{l}\right)_{ij} & \text{if } F_{ij}^{l}>0\\ 0 & \text{if } F_{ij}^{l}<0.\end{cases} \quad (2)$$

Feature correlations given by gram matrix:

$$G_{ij}^{l}=\sum_{k}F_{ik}^{l}F_{jk}^{l}. \quad (3)$$

The contribution of layer G to the total loss:

$$E_{l}=\frac{1}{4N_{l}^{2}M_{l}^{2}}\sum_{i,j}\left(G_{ij}^{l}-A_{ij}^{l}\right)^{2} \quad (4)$$

The total loss is:

$$\mathcal{L}_{style}(\vec{a},\vec{x})=\sum_{l=0}^{L}w_{l}E_{l} \quad (5)$$

Derivative of E_l with respect to the activations in layer l:

$$\frac{\partial E_{l}}{\partial F_{ij}^{l}}=\begin{cases}\frac{1}{N_{l}^{2}M_{l}^{2}}\left(\left(F^{l}\right)^{\mathrm{T}}\left(G^{l}-A^{l}\right)\right)_{ji} & \text{if } F_{ij}^{l}>0\\ 0 & \text{if } F_{ij}^{l}<0.\end{cases} \quad (6)$$

Minimize of loss function:

$$\mathcal{L}_{total}(\vec{p},\vec{a},\vec{x})=\alpha\mathcal{L}_{content}(\vec{p},\vec{x})+\beta\mathcal{L}_{style}(\vec{a},\vec{x}) \quad (7)$$

In order to build a style representation that compute the correlations between the difference filter response, we use Gram matrix to measure it as suggested by the original paper. We use gradient descent from a white noise image to find another image that matches the style representation of the original image, this is done by minimizing the mean-squared distance between the entries of the Gram matrix from the original image and the Gram matrix of the image to be generated. We define the loss function following the formula in the original paper.

In our experiment, just use the two loss function from the original paper is still not enough since the result will contain too much noise. In order to solve this problem, we add total variation as the penalty function. We reference lasange implementation of total variation penalty to write this function.

3.2.4 Choose optimization algorithm

At first we choose adam as our optimization algorithm, however, it takes too long time to generate some decent results. We use adam to perform some task of the original paper with low efficiency. In the CS231n course from Stanford, the professor talked about neural style transfer in the class. He mentioned that bfgs is better than adam in this task. Therefore we decide to try bfgs algorithm. We use scipy.optimize to implement bfgs and reference lasange code for the theano implementation. When we perform bfgs, it tooks significantly less time than adam to generate similar results. Therefore, we decide to use bfgs for most of our experiments.

3.2.5 Content and style reconstruction

For content reconstruction, in order to analysis the information in each layer, we reconstructing the image only from the feature maps in that layer.

For style reconstruction, we use the feature space originally designed to capture texture information as mentioned in the original paper. This feature space is built on top of the filter responses in each layer of the network. It consists of the correlations between the different filter responses over the spatial extent of the feature maps. We reconstruct the style of the input image from style representations built on different subsets of CNN layers, which is conv1_1, conv1_1 and conv2_1, conv1_1, conv2_1 and conv3_1, conv1_1, conv2_1, conv3_1 and conv4_1, conv1_1, conv2_1, conv3_1, conv4_1 and conv5_1. And we also reconstruct the image from the feature map in corresponding layer.

3.2.6 Style transfer

At last we perform style transfer combing the loss function of content and style. We modify the parameter alpha and beta several times and compare the results.

1. Provide system drawings, block diagrams, and/or circuit schematics for your software or hardware design, as applicable to your project;
2. Include flow charts and pseudo code descriptions for the step-by-step discussion of your software design.

4. Implementation

In this section we will present the Network design, training algorithms as well as data used of this project with details of the implementation.

4.1. Deep Learning Network

Algorithm

bfgs

the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm is an iterative method for solving unconstrained nonlinear optimization problems.

The BFGS method approximates Newton's method, a class of hill-climbing optimization techniques that seeks a stationary point of a (preferably twice continuously differentiable) function. For such problems, a necessary condition for optimality is that the gradient be zero. Newton's method and the BFGS methods are not guaranteed to converge unless the function has a quadratic Taylor expansion near an optimum. These methods use both the first and second derivatives of the function. However, BFGS has proven to have good performance even for non-smooth optimizations.

Adam:

Adam is variant on RMSprop+momentum with a few important distinctions. The momentum of Adam is incorporated directly as an estimate of the first order moment with exponential weighting of the gradient. The most straightforward way to add momentum to RMSprop is to apply momentum to the rescaled gradients (not particularly well motivated). Adam includes bias corrections to the estimates of both the first-order moments (the momentum term) and the (uncentered) second order moments to account for their initialization at the origin.

4.2. Software Design

Psudo code descriptions:

1. build neural network architecture
1.1 build a VGG19 network
1.2 remove the last three layer of original VGG19 network.
1.3 replace the max-pooling operation by average pooling

2. Preprocess the image
2.1 resize the image
2.2 cut the image
2.3 manipulating the axes

3. Define the loss function
3.1 define the content loss function
3.2 define the style loss function
3.3 define the total variation loss function
3.4 define the combination of these loss function

4. Choose optimization algorithm
4.1 try adam
4.2 try bfgs
4.3 compare the running time and results
4.4 choose the better algorithm for this task.
5. content and style reconstruction
for each layers mentioned in the paper:
    do the style and content reconstruction
discuss the results

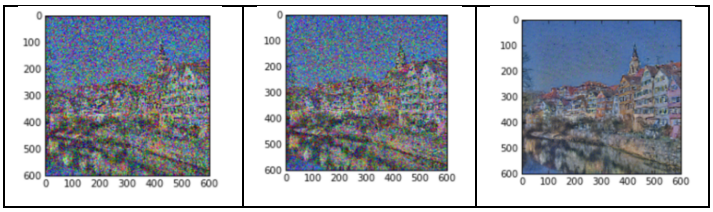
6. Style transfer
6.1 combine the loss function of style and content and perform training
6.2 experiment with different emphasis on style and content
6.3 discuss the results.

5. Results

5.1. Project Results

results:

5.1.1 total variation loss discussion

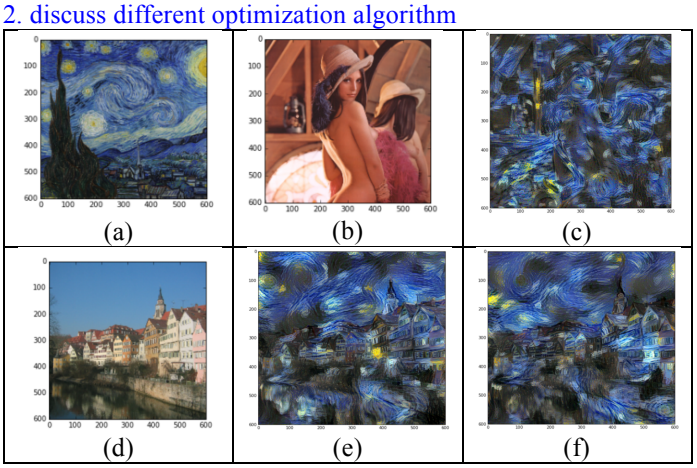


(a) (b) (c)

(a) style + content, no total variation penalty
(b) style + content, 1e-7 total variation penalty
(c) style + content 1e-5 total variation penalty

As we can see, without total variation loss adding to the loss function, the noise of the image is relatively large. In our experiment, we choose 1e-5 as the coefficient of total variation loss.

2. discuss different optimization algorithm

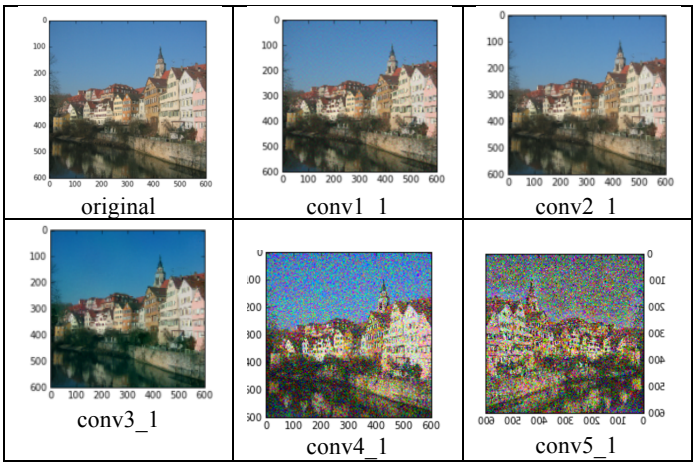


(a) (b) (c)
(d) (e) (f)

(a)(b)(d) original style and photo.
(c)(e) results using adam
(f) result using bfgs

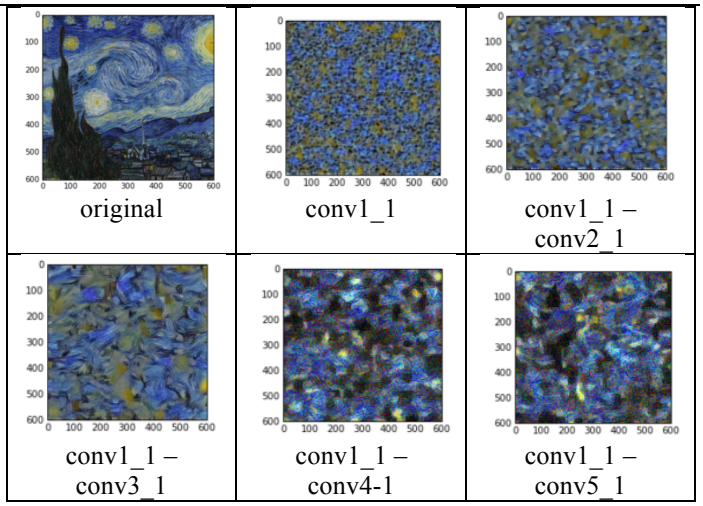
As we can see from the picture above, both adam and bfgs can generate good results (this result put emphasis on style, you can zoom out the picture to see the detail of house or lena). However, the run time of this two algorithm is very different. Bfgs is much faster than adam.

3. content reconstruction



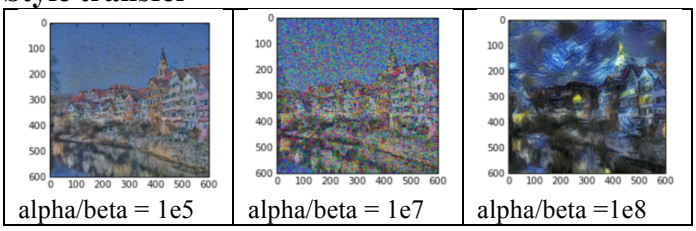
As we can see from the result, the higher layer's content reconstruction is less vivid than the lower layer.

Style reconstruction



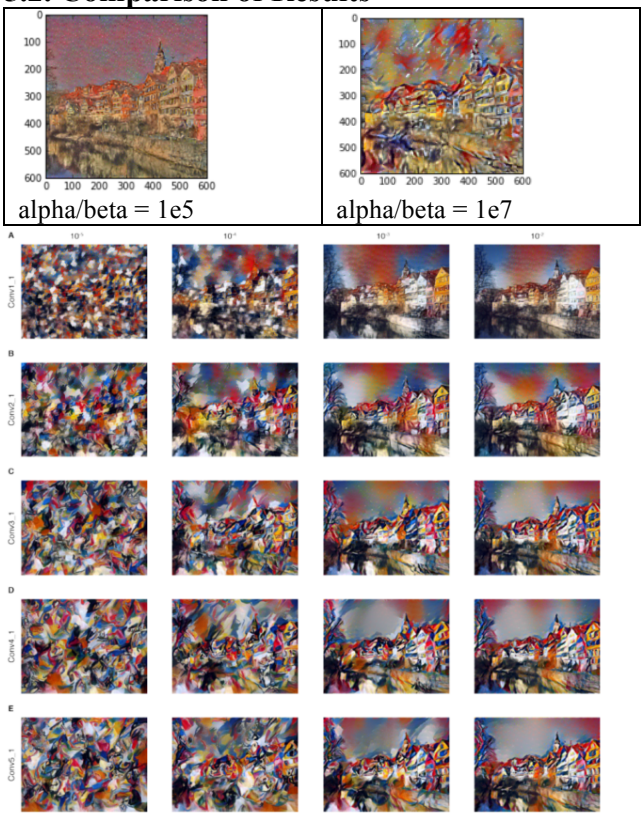
As we can see from the results, the lower layer can't reconstruct style greatly, but with more convlayers, the style can be reconstructed.

Style transfer



As we can see from the result, we can modify the emphasis on style and content by modifying the alpha/beta

5.2. Comparison of Results



As we can see from the result, in our experiment the alpha/beta is different from paper when generating similar results. It might because the different implementation and such settings such as total variation loss.

6. Conclusion

In this project we implement a neural algorithm with Theano to explore artistic style transfer by performing style and content representation respectively and then combine them together, during the process of which we discuss the results with different parameter settings as well as optimization approach.

6. Acknowledgement

Thanks for Professor Zoran Kostic, for his lectures and tutorship and thanks for all TAs for their help during the project.

Thanks for Professor Zoran Kostic, for his lectures and tutorship and thanks for all TAs for their help during the project.

8. Appendix

8.1 Individual student contributions - table

	CUID1	CUID2	CUID3
Last Name	Xx2241		
Fraction of (useful) total contribution	1/3	1/3	1/3
What I did 1	The main part implementation		
What I did 2	The loss function		
What I did 3	The main part implementation		

8.2 If/as needed: additional diagrams, source code listing, circuit schematics, relevant datasheets etc.