A Neural Algorithm of Artistic Style
E4040.2017Fall.WZYD.report
Zhanlue Yang zy2284, Shuizhou Wang sw3192, Xiaoxiang Zhang xz2631
Columbia University

Abstract
In this project, we analyze the performance of different networks and test the effect of inserting batch normalization layers into VGG network. We build three types of network in this project. After some testing, parameters setting is done and the balance between content loss and style loss is achieved.

1. Introduction
Capturing the style of images and transfer it onto another image is a kind of texture transfer problem. In these years, we have witnessed many impressive progresses in this area both in non-parametric methods[2-4] and in neural network[1]. Besides, Manuel Ruder proposed a method which can transfer the style from one image to a whole video[5].

In [1], author separates and reassembles content features and style features by using Deep Neural Network to extract high-level feature representations from VGG network. This project extents the results from that paper. Besides VGG, we built three types of networks, which are VGG+ batch normalization, Resnet-20 and Resnet-56, and tried to analyze the performance differences among them. Besides, we analyzed the performance difference after we inserted normalization layer into VGG.

Through this project, we did many times of experiments to adjust parameters, such as learning rate, iterations, the choice of feature layers and loss parameters, in order to balance and minimize the content loss and style loss. At the end, we can construct the graph with both content loss and style loss smaller than 1.

2. Summary of the Original Paper
2.1 Methodology of the Original Paper
This paper introduces a system which can separate and recombine content and style of arbitrary images via Deep Neural Network. It can manipulate the appearance and content of images independently to produce new meaningful images.

This system is built based on VGG-Network. More specifically, authors used 16 convolutional and 5 pooling layers of the 19-layer VGG. There is no fully connected layers in the network. Besides, compared to max-pooling operation, author prefer to use average pooling instead because they thought it can improve the gradient flow. The detailed method is by performing gradient descent on a white noise image and trying to minimizing the loss. The loss function is composed by the sum of squared-error loss of feature representations and Gram matrix (A matrix comprising of correlated features).

The graph of VGG model used and the flow chart are posted below.

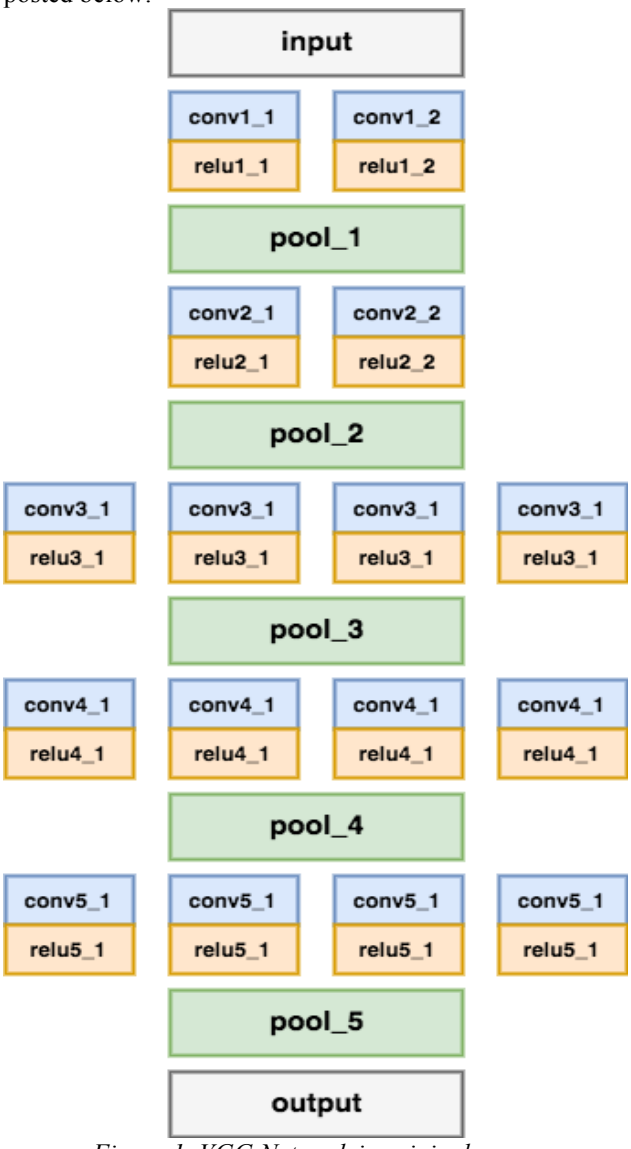


Figure 1. VGG Network in original paper

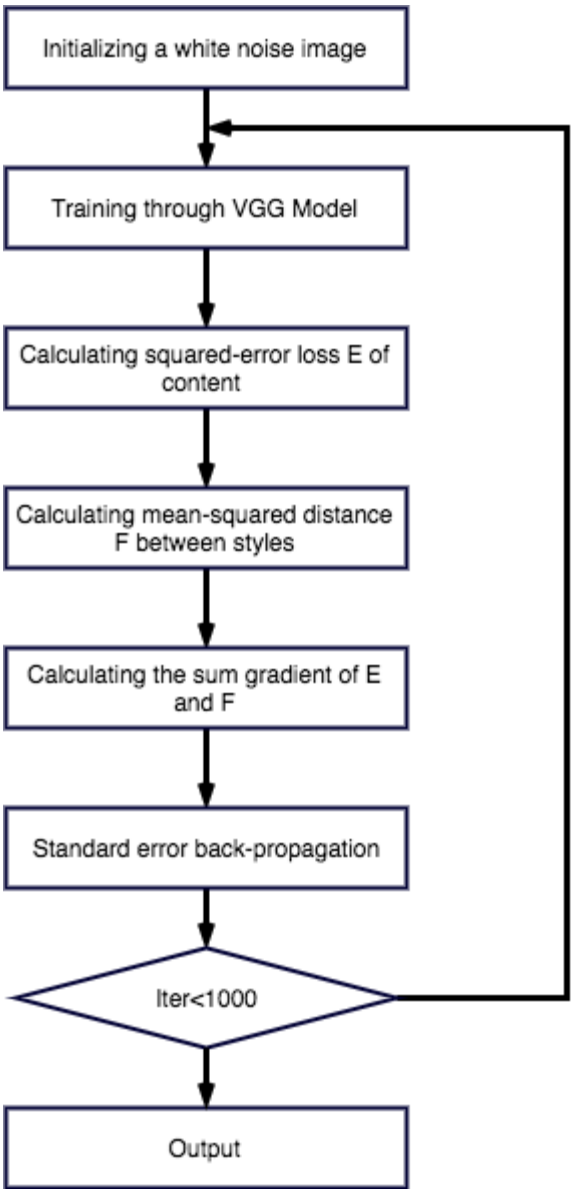


Figure 2. flow chart for VGG Network in original paper

2.2 Key Results of the Original Paper

As been discussed in the paper, authors proposed a new fascinating and effective method on synthesising images which uses Deep Neural Networks to retrieve high level content and complex-cell like computation to derive the appearance (styles). This result is mainly based on the finding that the representations of content and style in the Convolutional Neural Network are separable, which means that they can be manipulated independently to produce new images. In detail, the key results illustrated in this paper are listed below:

1. Along the processing hierarchy of the network, the input image is transformed into representations that increasingly care about the actual content of the image compared to its detailed pixel values.
2. The style representation is derived by computing correlations between the different features in different layers of the CNN.
3. The local image structures captured by the style representation increase in size and complexity when including style features from higher layers of the network.
4. A transformation into stationary feature space might achieve excellent performance in style classification.
5. Performing a complex-cell like computation at different processing stages along the ventral stream would be possible way to obtain a content-independent representation of the appearance of a visual input.

3. Methodology

3.1. Objectives and Technical Challenges

Basically, the objective for this project is to combine the style and content characters from two different graphs to create a whole new graph.To achieve our goal,There are some possible challenges.

1. During the process of parameter adjustment, the proportion composition of the content coefficient ($cont_{coeff}$) and style coefficient ($style_{coeff}$) is hard to determine. Therefore, it takes time to adjust these parameters so as to create an appealing output picture. When synthesising an new image that combines both the style and content features from original pictures, there often does not exist a picture that fit in both of them perfectly at the same time. As a result, it is important to find out a wise strategy to seek a proper balance between these two constraints according to the specific neural network structure. For example, we could apply typical Control Variable Method (CVM) in the experiment: we will concentrate on figuring out the appropriate proportion of content loss and style loss in order to reach an optimal loss function which can be used in training model.
2. Construct test model with suitable layers to maximize efficient features form style and content pictures. In this way, we could extract features which can affect our model and results. Also, we are supposed to construct an effective network structure. For example, we need to make the decision between a higher or a lower layer. Moreover,we need to examine the concrete effects of extracting features from different types of layers.
3. Deal with the unexpected problem in building a entire new neural network. Apart from utilizing

the VGG-19 the original paper has discussed, we are willing to fully apply and understand the deep learning issues conceptually and practically by designing and constructing an entire new neural network to solve the problem proposed at the beginning.Thus, there are multiple uncertainties and difficulties we may face in the following research.

4. Deep understanding of the original VGG-19 and conduct effective optimization. We need to analyze each section of the original neural network structure and the relationships among different parts of the network. It can be quite time-consuming to test and analyze the inner-relationship existing in the network., Therefore, we need to conduct our analysis effectively and move on to realize optimization in VGG-19. From now on,our startpoint is to add a batch normalization layer to the initial network learned from paper and to change our pooling layer method.

3.2. Problem Formulation and Design

In this project, we use three types of neural network to combine content and style from different graphs. We calculate the loss from white noise, content and style and use gradient to minimize it.

3.2.1 Mathematical Protocol and Engineering Design

Different from the system in the paper, we used a modified VGG-19 network in this project, which contains a batch normalization layer after each convolutional layer. This model was built in Tensorflow framework. The basic design conceptual and technical methods can be illustrated from the graph below.

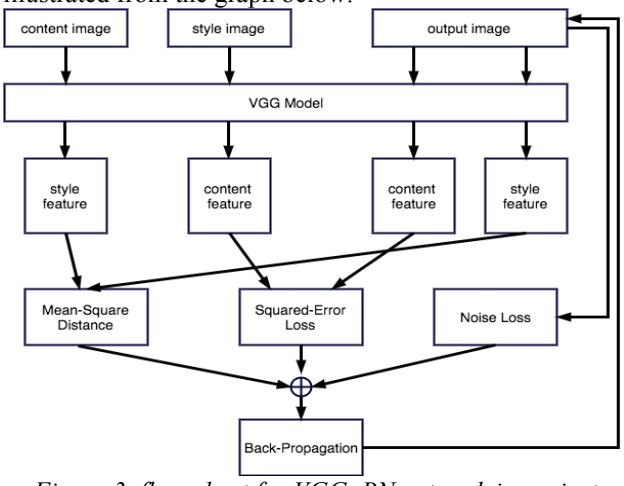


Figure 3. flow chart for VGG_BN network in project

To build a whole new graph using the content feature and style feature from existing graphs, the loss from three aspects need to be minimized, which are the style loss (the mean-squared distance between style features), content loss (the squared-error loss between content features) and the noise loss (which measures the pixel wise smoothness). After we have derived the sum of these three losses, we can do back-propagation to update the output graph.

In the following part, I am going to illustrate this process step by step:

1. Use CIFAR 10 to train the VGG network in the first place.
2. Use already trained VGG network to derive content features and style features from input graphs and white noise graph.
3. Calculate the squared-error loss between the content features from input graph and the white noise graph. The protocol can be derived as below, in which *l* represents the layer number, *i, j* represents the filter's number and the future's number.

$$E = \frac{1}{2}\sum_{i,j} \left(F_{ij}^{l} - P_{ij}^{l}\right)^2$$

4. Then we need to derive the loss function for the style feature, which are measured by the correlation between different groups. This can be achieved by using Gram matrix G, which presents the inner product between vectorised featured map. For example, G_{ij}^{l} represents the inner product of filter *i* and filter *j* in layer *l*. Thus the loss for style feature can be represented as:

$$F = \frac{1}{4N_l^2 M_l^2}\sum_{ij} \left(G_{ij}^{l} - A_{ij}^{l}\right)^2$$

in which G and A are the style feature maps.

5. Besides, for the noise loss N, we need to use the L2-norm to minimize it.
6. The total loss for this system is a weighted sum for these three parameters, which can be represented as:

$$L = \alpha E + \beta F + \gamma N$$

7. By doing back-propagation on the total loss, we can update the output graph to let it be more similar to the content graph and style graph.
8. Repeat this process for a certain iterations, which is a parameter been set in our program.

3.2.2 Software Design

Based on the neural style model in the paper, we built three types of models in this projects, which are VGG_BN, Resnet-20, Resnet-56 respectively. For the VGG model, we added a batch normalization layer after each convolutional layer to accelerate convergence process and called this modified model VGG_BN. The code for Resnet-20 and Resnet-56 models are completely coded by ourselves based on original VGG code.

pseudo code：

```
for(i< 1000):
        train network via Cifar10 database
```

```
        if(result is better than the best result):
                best result= result
for(i< 1000):
        extract content and style features from inputs
        calculate the sum of loss
        back-propagation updates the results
```

4. Implementation

In this section, we are going to introduce the detailed design of deep learning network and the software design.

4.1. Deep Learning Network

We inserted batch normalization layers after each convolutional layer to build our VGG_BN networks. By adding normalization layers, we can have faster convergence speed compared to normal VGG network.

The structure for Resnet-20 and Resnet-56 are composed by convolutional layers, max-pooling layers and batch normalization layers, whose structure is quite well-known.

Basically, there are two kinds of training process in this project. The first one is by using Cifar10 to train the network. The second one is to minimize the content loss and the style loss by verifying the variables.

The database we used in this project is Cifar10. And we also use some graphs as the source of content features and style features.

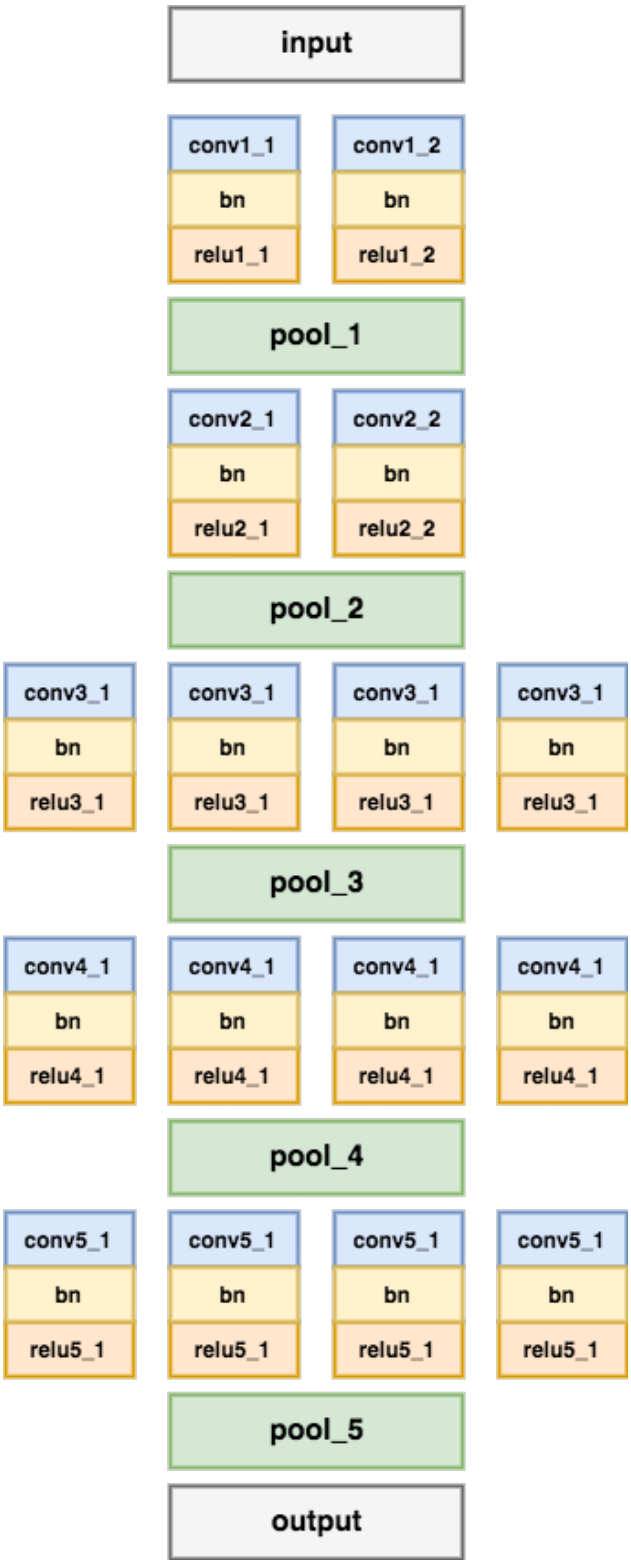


Figure 4. VGG_BN Network in this project

4.2. Software Design

The whole software design can be divided into three parts. We are going to introduce them one by one.

4.2.1. Model Training Process

The training process is quite similar between VGG network and Resnet, which can be seen from the graph below.

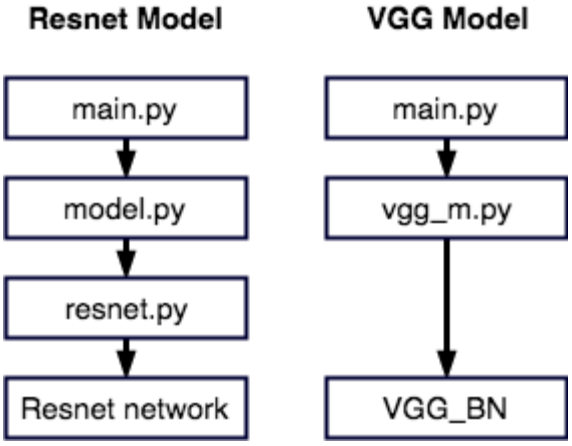


Figure 5. flow chart for training process

The specific function for each python file is described in this part:

1. **main.py:** Contain the main part of the training process, which set the important parameters for this process. Such as, it sets that the batch-size is 256, optimizer is Adam and the learning rate is 0.001. We used Cifar10 database in this project and did some comparison between their results. Besides, we set that there would be a shuffle after each echo.
2. **resnet.py:** Define all types of basic layers will be used in Resnet Neural Network. Such as convolutional layer, pooling layer.
3. **model.py:** Combine basic layers together to get the Resnet network. We can simply switch between Resnet-20 and Resnet-56 by changing the number of layers, which is an input for this file.
4. **vgg_m.py:** Basically, it's a combination of resnet.py and model.py. It defines all basic layers and combine them together as a VGG_BN network.

The detailed flow chart for Resnet network is displayed below. Situation for VGG network would be quite similar with it.

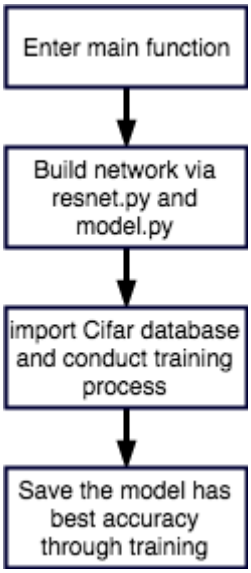


Figure 6. flow chart for resnet network

Note that for the best accuracy model, we only save the variables before fully-connected layers.

4.2.2. Tensorboard Analysis

The python files of tensorboard analysis for Resnet and VGG networks are *image_show_vgg.py* and *image_show_resnet.py*. We can retrieve all feature maps through this process and use them to measure the content-extraction ability and style-extraction ability for this model. The detailed method is:

1. Build a network which is same as the training network
2. Import the input graphs and the best accuracy model from training process
3. Save feature maps from all the layers via *tf.summary* function
4. View those graphs via tensorboard.

4.2.3. Neural Style Transfer

We combined content features and style features to construct a new graph in this part.

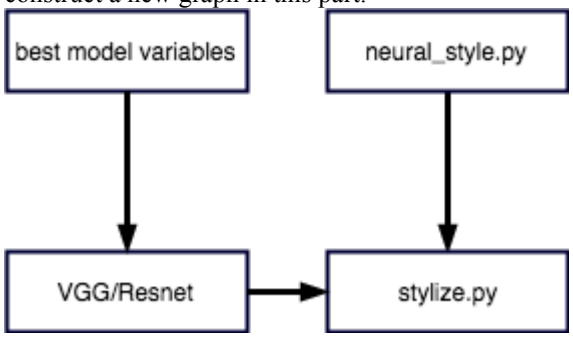


Figure 7. flow chart for neural style transfer

neural_style.py provides the entry for this process. We import content image and style image through this python file. Besides, we import the best model variables to the network we used as well. stylize.py is the main function for this process. Both the input images and best model

would be imported into this file and conduct construction process. The detailed process can be concluded as:

1. Collect input information and pass them to the stylize.py file
2. stylize.py would build two types of models which take content image and style image as input respectively and use variables in best model to construct the network.
3. Extract content-feature and style-feature from these two network and store them as two *np.ndarray* array in large_list. Besides, we also evaluate all variables in pre-trained model at the same time.
4. Transform all tf.variable into tf.constant and use them to build the third network.
5. Import graphs as content-size tf.variable and optimize it via Adam. Finally, output the graph when transformation is done.

4.2.4. General Flow Chart

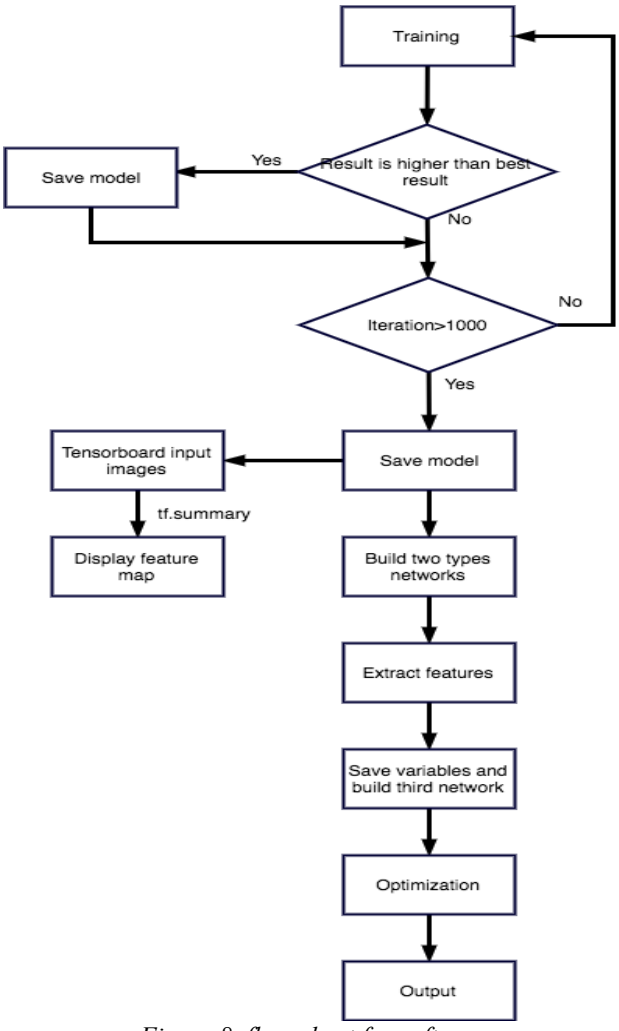


Figure 8. flow chart for software

4.2.5. Pseudo code

Training:

1. Build Network (VGG_BN, Resnet-20, Resnet-56)
2. Set Parameters: Learn Rate, Beta1, Beta2, Gamma, Initialization, etc
3. Define loss
4. Set Optimizer: AdamOptimizer
5. In each epoch:
   shuffle Training Images
   traverse entire training dataset:
   run optimizer to minimize loss, update weights/bias
   compute accuracy
   if best_accuracy:
   save model to file

Feature Map:

1. Build Network, using the same structure as training network
2. Restore weights/bias from saved model
3. Load image
4. Run tf.summary to record 'event' file
5. Use command 'tensorboard –logdir=xxx' to obtain feature map

Neural Style Transfer

1. Build Two Networks with the same structure as training network
2. Create input placeholder for each network, with style image shape and content image shape, respectively
3. Compute content feature and style feature, then save in np.ndarray format.
4. Evaluate variables in either of the two networks, save them to Layer_List in np.ndarray format
5. Create Third Network with the same structure as training network, but with all variables replaced by tf.constant with values obtained from Layer_List.
6. Create input placeholder for the third network, using tf.variable with content image shape.
7. Define content_loss = content_feature – content_variable
8. Define style_loss = style_feature – style_variable
9. Define tv_loss to suppress random noise
10. Define total loss = style_loss + content_loss + tv_loss
11. Set Optimizer: Adam
12. Initialize the input variable
13. Train for 1000 iterations
14. Evaluate input variable as output image

5. Results

5.1. Project Results

5.1.1 analysis method

Basically, there are four neural networks involved in this project: The first neural network is the traditional neural network named VGG_ORIG which is examined in the paper we reviewed at the beginning. The second neural network is called VGG_BN which is constructed based on the modification of the original VGG_ORIG from paper. There are two more networks designed and constructed named Resnet20 and Resnet56 respectively. Thus,we conducted a series of experiments using all these four networks. In this way, we managed to analyze the overall performance of these different networks. Particularly, the effect of our addition of the Batch Normalization layer in our networks was examined. The content picture and style picture we used for results demonstration can be seen below in Fig. 9

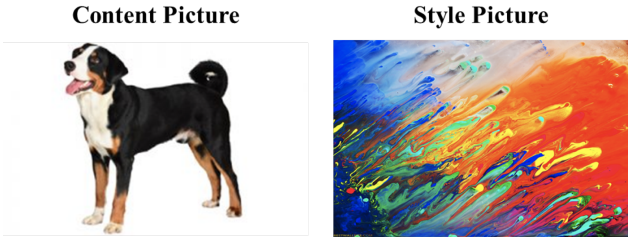


Figure 9. content graph and style graph

5.1.2 best model results

The best results obtained from these four networks respectively are presented as below in Fig. 10.

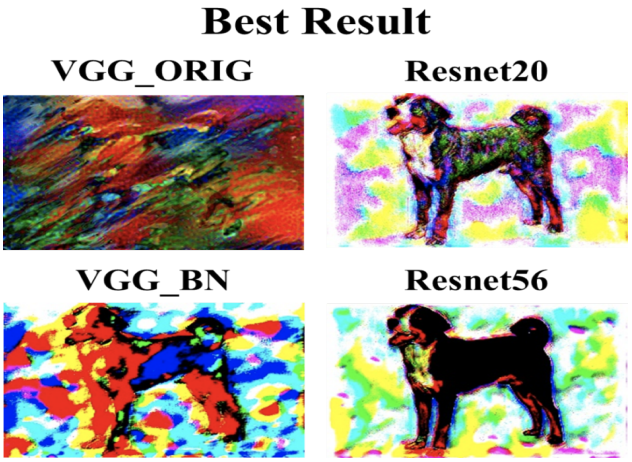


Figure 10. Optimized neural style transformation result for Original VGG, VGG_BN, Resnet-20 and Resnet-56

5.1.2 analysis results

The best transformation result after parameters optimization has been shown for each individual networks. We will give more details about our results of each parts in the 5.2 part, now we will conclude our general results here. As discussed in the following passage, the VGG_BN network showed a well balance between style and contents, while Resnet 20 and Resnet 56 remained very little original styles. On the contrary, original VGG network exhibited over-strong style extraction, thus remained little original content features.

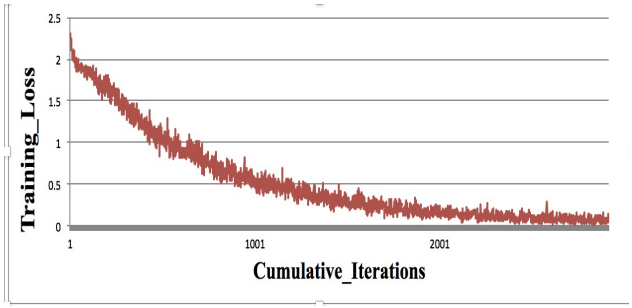
5.2. Comparison of Results

5.2.1 analysis method illustration

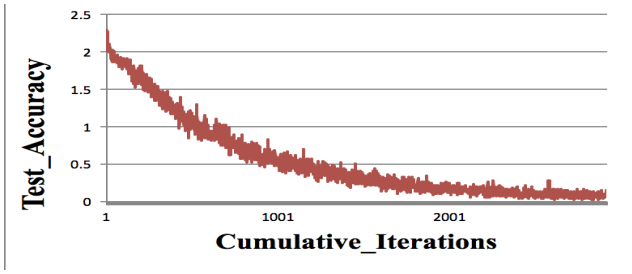
In order to analyze the performance of different networks, as well as the effect of inserting a Batch Normalization Layer, we performed the following experiments for each of the networks (VGG_Orig, VGG_BN, Resnet20, Resnet 56).

1. test error and train error illustration

According to our specific topic, we'd like to gain an appealing picture in our project. Thus, we'd like to to show the picture feature effect visually.Thus**,** we concentrate on testing our model performance by applying method below. Regarding the test error and train error in our process of training, we can get an intuitive result as below:



Plot. 1 train loss with iterations



Plot 2. test accuracy with iteration

2. loss function control.

Traditionally, the loss for neural style transformation is calculated using formula as shown below:

$$loss = style_{loss} + content_{loss} + tv_{loss}$$

However, in this project, we used a modified version of formula instead as shown below. $loss = style_{coeff} * style_{loss} + cont_{coeff} * content_{loss} + tv_{coeff} * tv_{loss}$

We adjusted this loss function primarily to make comparisons for individual ability. In this way, by tuning the percentage of $cont_{loss}$, $style_{loss}$, and tv_{loss}, we could easily adjust their respective corresponding coefficient values, so as to analyze the balance among all these three losses.

3. strategies of analysis.

Here are some of the analysis methods we used in our project:

1). Feature Map Analysis:

The tf.summary function was utilized in the process of constructing the graph. Also, we used tensorboard to obtain the output which is the feature map of each output layer.

2). Extraction Ability Analysis

We use learn_rate = 100 for analyzing all extraction abilities. We test three kinds of extraction ability: Current Extraction Ability, Style Extraction Ability and Tv_loss Effect to analyze the information extract ability in different neural networks and layers.

For testing content extraction ability, we set style_coeff = 0 and tv_coeff = 0; For testing style extraction ability, we set cont_coeff = 0 and tv_coeff = 0; For testing tv_loss effect, we set style_coeff = 0 and cont_coeff = 0

3). Parameters Adjustment:

We further adjust the parameters of style_coeff, cont_coeff, tv_coeff, and learn_rate to obtain an optimum neural style transformation effect.

5.2.2 comparison results

1.The comparison on different layer

a.vgg_bn:

1). content extraction ability from lower layer to higher layer:

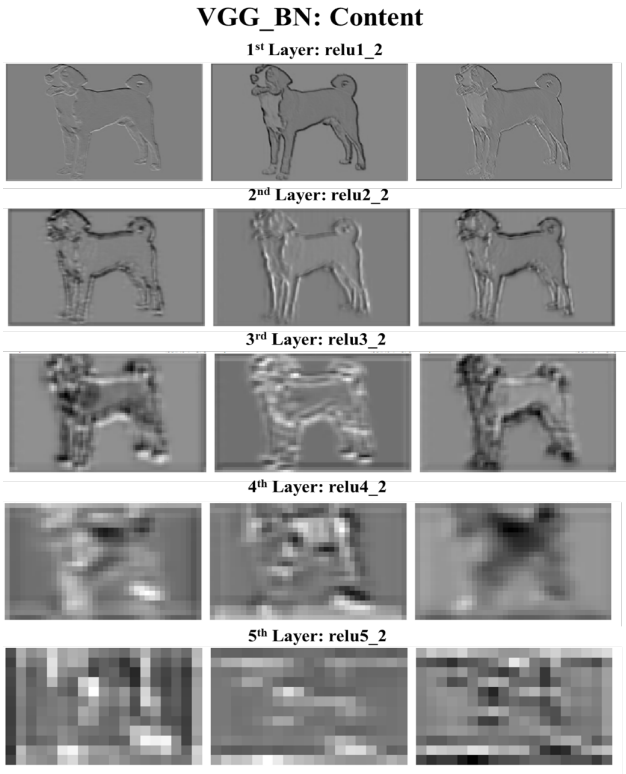


Fig. 11 Feature map for VGG_BN using content picture as input

As shown above, the content extraction ability of first 3 layers in VGG_BN is pretty strong. However, in the 4th layer, we can only see very ambiguous features of content, which nearly vanished when moving forward to the 5th layer.

2). style extraction ability and Tv_loss Effect.

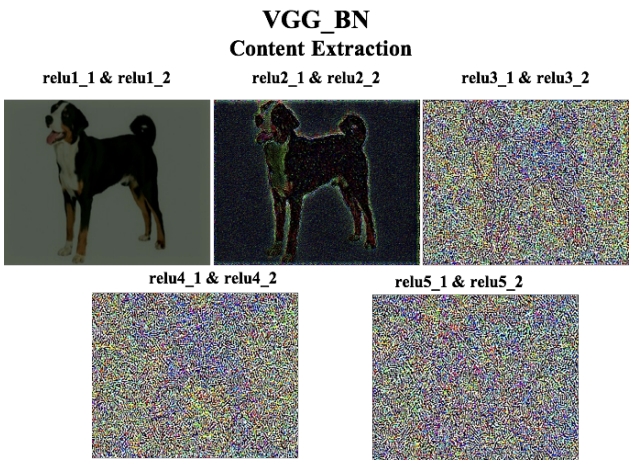


Figure 12. Content Extraction, Style Extraction and Tv_loss Effect analysis of VGG_BN network

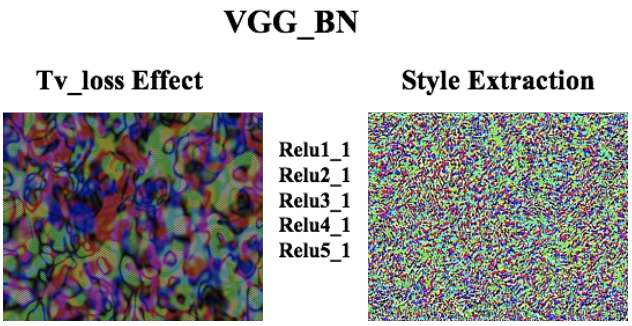


Figure 13. Style Extraction and Tv_loss Effect analysis of VGG_BN network

As shown above, we can see the Content Extraction, Style Extraction and Tv_loss Effect of VGG_BN network.

It is apparent that deeper layers in VGG_BN exhibits a relatively poor recovery of original contents. While VGG_BN exhibited low ability in style extraction, tv_loss in VGG_BN shows a strong ability of extracting color blocks from the style graph. However, no style features can be extracted from tv_loss.

3).transfer effect when using different output form different layers

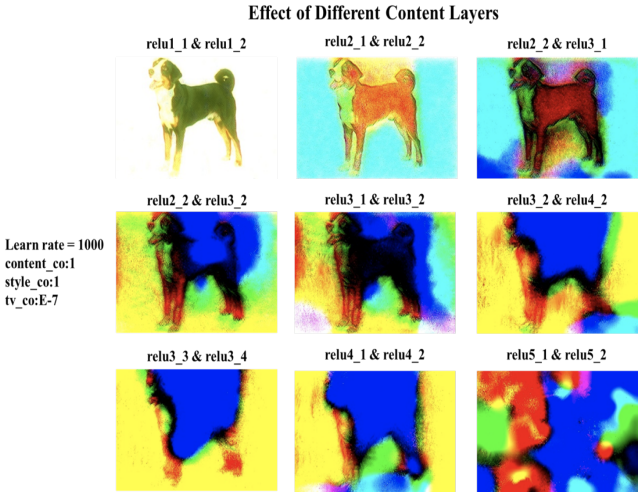


Figure 14. Effect of using different layers of VGG_BN as content layers

As shown above, the actual neural-style transfer effect when using output from different layers in VGG_BN. The final transformation effect is a balance between content_loss, style_loss, and tv_loss, as can be clearly seen from Fig. 14. While the first two layers (relu1_1, relu1_2, relu2_1, relu2_2) exhibited over-strong content extraction ability, we tend to use relu3_1 and relu3_2 as content layers, as they achieved a pretty balanced result between style and contents.

4).optimal parameters adjustment

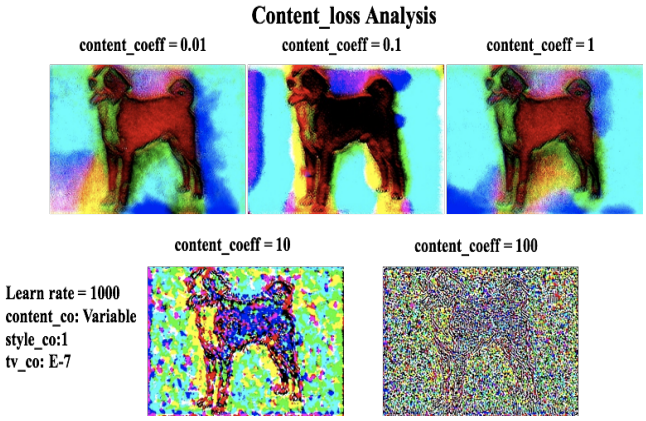


Figure 15. Analysis of content_coeff for VGG_BN

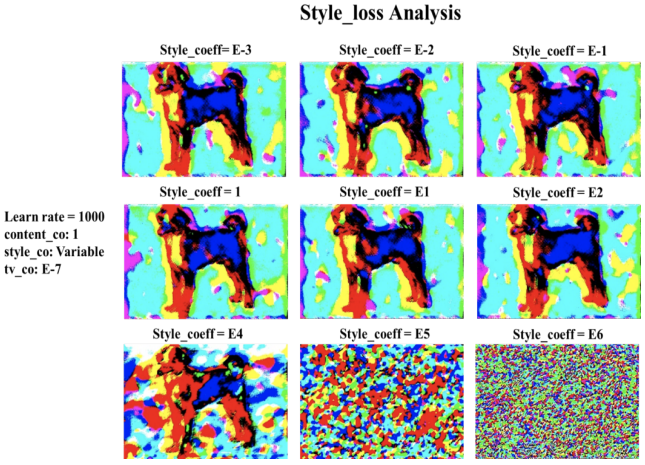


Figure 16. Analysis of Style_coeff for VGG_BN

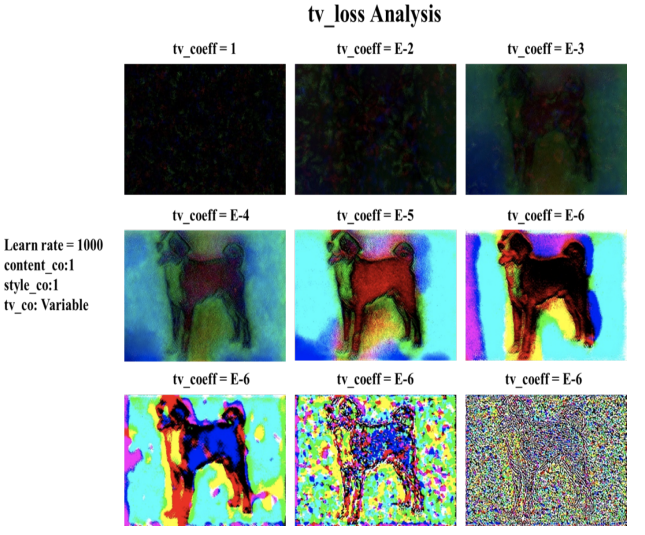


Fig. 17. Analysis of tv_coeff for VGG_BN

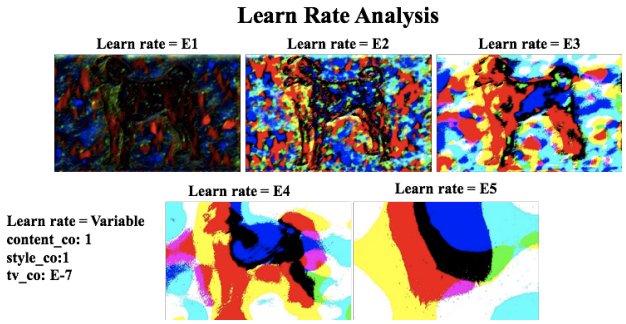


Figure 18. Analysis of learn_rate for VGG_BN

Figure 14 – Figure 18 shows parameters adjustment for content_coefficient, style_coefficient, tv_coefficient, and learn_rate, with the ultimate goal of achieving optimized balance or performance between different losses. **Optimized parameters** are: lr = 10000, content_coeff = 1, style_coeff = 10000, and tv_coeff = 1E-7, with relu3_1 and relu3_2 as content layers, relu1_1 – relu5_1 as style layers.

2. The comparison on different network

a.VGG_BN & VGG_ORIG:

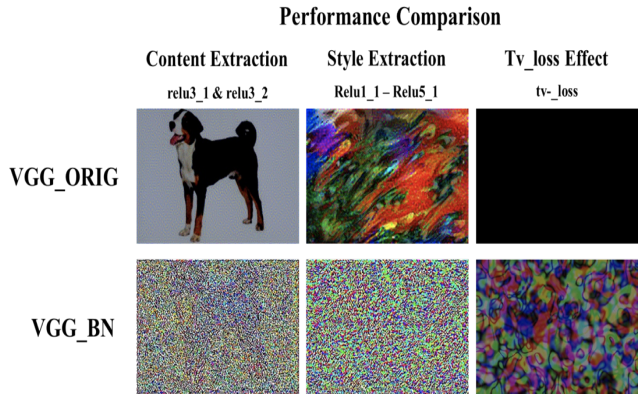


Figure 19. Comparison between Original VGG and VGG_BN networks

As a comparison with original VGG-19 network, Fig. 19 shows that the addition of BN layer would reduce the ability of both content extraction and style extraction. However, with the existence of BN layer, tv_loss would obtain a novel ability of extracting color blocks.

b.VGG_BN & Resnet20, & Resnet56:

VGG_BN&Resnet20:

1). content extraction ability

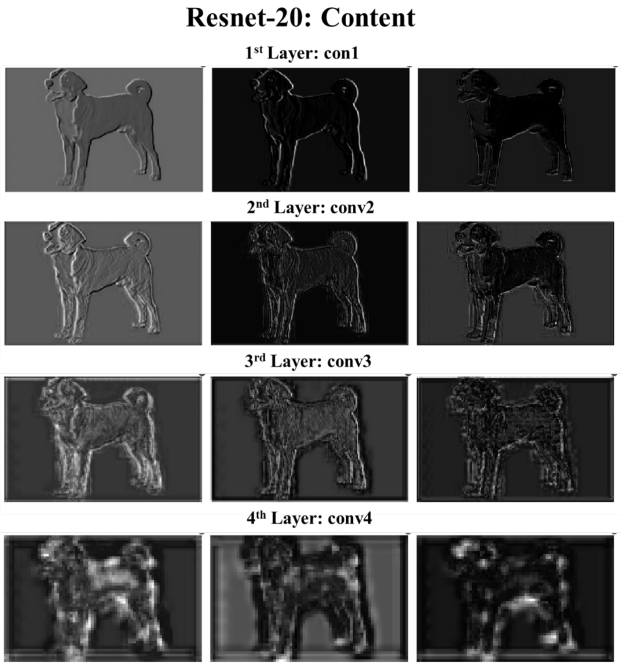


Figure 20. Feature map of Resnet-20 using content picture as input

Figure 20 shows the feature map of each layer in Resnet-20.This feature map indicates the strong ability of content extraction of Resnet-20. Unlike VGG_BN, even in the last layer of Resnet-20, the features of content is still very clear.

2).style extraction ability, tv_loss effect

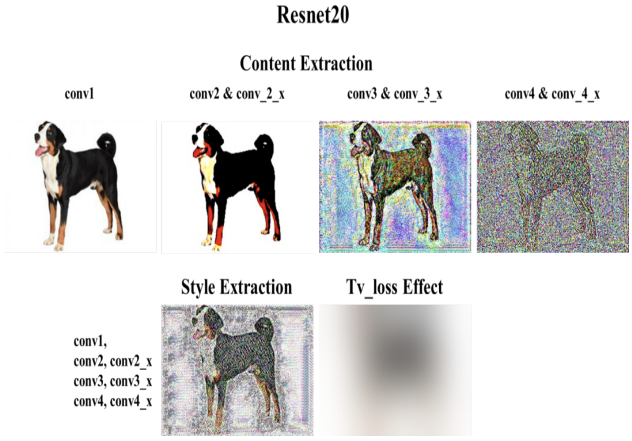


Figure 21. Content Extraction, Style Extraction and Tv_loss Effect analysis of Resnet-20 network

Resnet 20 exhibited an extraordinary strong ability of content extraction, as shown in Fig. 21. Unlike the vgg we discussed in the former passage, even the last two layers

(conv4, conv_4_x) exhibited a strong ability of content extraction. Moreover, the style extraction can also be able to recover the content features. Tv_loss seems to play a relatively low effect in Resnet 20, only making the graph blur.

VGG_BN&Resnet56:

1).content extraction ability

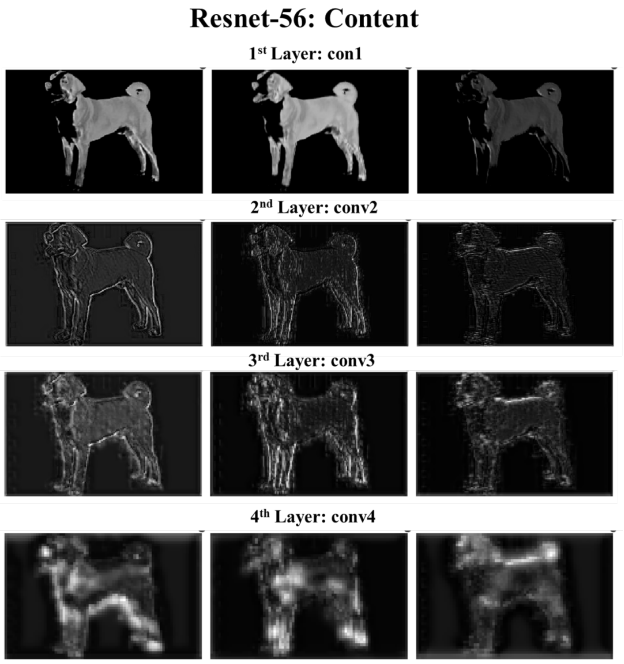


Figure 22. Feature map for Resnet-56 using content picture as input

As shown above, similar to Resnet-20, Resnet-56 also exhibits a strong ability of content feature extraction, which is better than VGG_BN.

2). style extraction ability, tv_loss effect

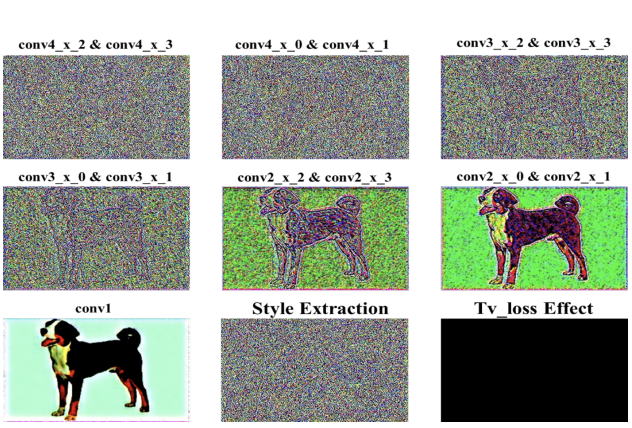


Figure 23. Content Extraction, Style Extraction and Tv_loss Effect analysis of Resnet-56 network

As shown above, while Resnet-56 performs better in content feature extraction.unlike VGG_BN, the tv_loss in Resnet 56 could not extract color block*s*.

Resnet20&Resnet56:

As figures show above, compared with Resnet 20, Resnet 56 showed a poor ability of both content and style extraction.

overall comparison:

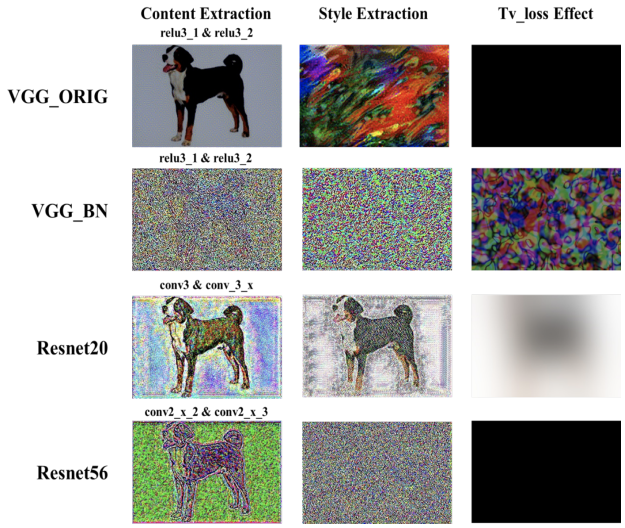


Figure 24. Content Extraction, Style Extraction and Tv_loss Effect comparison among Original VGG, VGG_BN, Resnet-20, and Resnet-56 networks

As a general conclusion, the original VGG network exhibits both strong content extraction and style extraction abilities, thus make obtain optimum performance in neural style transformation tasks. Aside of original VGG network, Resnet 20, Resnet 56, and VGG_BN shows different advantages and disadvantages.

VGG_BN shows relatively normal style extraction ability, but weak in content extraction, and the most unique ability of VGG_BN lies in the fact that its tv_loss could be applied for extracting color blocks in addition to noise suppress.

Resnet 20 shows an extraordinary content feature extraction, even the style loss could be able to recover the original content features. However, the style extraction, on the other hand, is pretty poor. Tv_loss in Resnet 20 makes the graph blurry, in addition to noise suppress.

Resnet 56 shows normal content extraction ability,which is better than VGG_BN but worse than Resnet20. Moreover, it behave a poor style extractions.

5.3. Discussion of Insights Gained

Basically, in terms of analysis process, we have conducted two types of analysis method. The first method is to discuss the performance of different layers within one network. The second method is to discuss difference of the overall performance of these four networks.

When it comes to the concrete discussion about the performance of these four networks, we can conclude that they all have their own positive and negative features. First, for the traditional VGG network, it can extract features from both the content image and the style image to a reasonably good extend. Second, for the modified version of VGG network (we call it VGG_BN), it could extract the style image feature to a normal level. However, it has relatively weak extraction ability of content image feature. Its unique strength is that its tv_{loss}could be utilized to extract color blocks apart from suppressing the noise. Third, for the Resnet20 network, it has fairly strong extraction ability of obtaining features from content image. However, one weakness of it is that its tv_{loss}parameter could not be applied to extract color blocks as what we could do using VGG_BN. Finally, for the Resnet56 network, it can be intuitively viewed as a 'middle' status between VGG_BN network and Resnet20 network. We are saying this because it has relatively stronger ability of extracting content image feature compared with VGG_BN, while has relatively weaker ability of extracting style image features compared with Resnet20 network at the same time.

6. Conclusion

Reflecting on the whole process of this project, we got improved both from theoretical knowledge understand and from practical use of knowledge we learned.

First, we reviewed some related previous work and went deep into the paper about traditional VGG network. We obtained a thorough understanding of it by conducting the parameter adjusting during a series of experiments. After that, we modified the traditional VGG network by primarily adding a batch normalization layer into it. We compared our modified version of VGG network with the original VGG network from various aspects. Then, to further understand the related deep learning knowledge and apply theory into practice, we designed and constructed two more networks called Resnet20 and Resnet56 respectively. Finally, we compared all these four networks from different perspectives mainly by examining the relation between network performance and content coefficient, style coefficient and tv_loss value.

In terms of the concrete experiment process, we primarily utilized the Control Variable Method (CVM) in the lengthy parameter optimization process. After a series of trails and fails as well as multiple times of discussions about the results, we found out the optimal parameter setting so as to finally obtain the "best model" for all these four networks respectively.

6. Acknowledgement

First of all, I would like to extend my sincere gratitude to my teacher, Prof. Zoran Kostic, for his patient introduction when we were choosing the research topic and his nice guidance in this semester.

Then, high tribute shall be paid to Tingyu Mao, Jingyu Qian, Pablo Vicente Juan, Abhishek Jindal, Wenyu Fu, Abhishek Bhatia, who is the TAs for this course. I truly appreciate for all their effort from patiently answering our questions and give us hints on how to improve our project. Without their generous help, we can't learn such much from this project.

7. References

Include all references - papers, code, links, books.

[1]Shuizhou Wang:
git@bitbucket.org:ecbm4040/2017_assignment2_sw3192.git
overall code:
https://bitbucket.org/neural_style/neu_improve

[2] Hertzmann, A., Jacobs, C. E., Oliver, N., Curless, B. & Salesin, D. H. Image analogies. In Proceedings of the 28th annual conference on Computer graphics and interactive techniques, 327–340 (ACM, 2001). URL http://dl.acm.org/citation.cfm?id= 383295.

[3] Ashikhmin, N. Fast texture transfer. IEEE Computer Graphics and Applications 23, 38–43 (2003).

[4] Efros, A. A. & Freeman, W. T. Image quilting for texture synthesis and transfer. In Proceedings of the 28th annual conference on Computer graphics and interactive techniques, 341–346 (ACM, 2001). URL http://dl.acm.org/citation.cfm?id=383296.

[5] Ruder M, Dosovitskiy A, Brox T. Artistic style transfer for videos[C]//German Conference on Pattern Recognition. Springer International Publishing, 2016: 26-36.