Universal Style Transfer via Feature Transforms

E4040.2018Fall.ICCV.report
Tingyu Guan tg2663, Hong Yan hy2557, Yimeng Zhang yz3397
Columbia University

Abstract

This project is to extract the style from any image and apply it on the other images, which could overcome the inability of feed-forward method. The main challenge is how to extract image styles and reconstruct the other images in specific styles. As the result of this project, high-quality images with specific styles are generated.

1. Introduction

One of modern-art creation methods is to transfer the styles of existing images. For example, there are two images, which are named the style image and content image, respectively. And style transfer is to obtain the style of the style image and apply this specific style to the content image. The main challenge of this project is how to extract the effective style representations of any images and then how to combine this style with the content of the other images efficiently. Existing pre-trained feed-forward based method [3,5,8,9,10] is inspiring. It uses Gram matrix [6,7] which is able to obtain visual styles remarkably. However, the limit of this existing method is that it cannot process new images to obtain ideal visual results. This project aims to find a simple but effective method to break this limitation. The model design for this project would consist of three main parts, which are encoder, whitening and coloring transform (WCT) layer, and decoder, respectively.

2. Summary of the Original Paper

2.1 Methodology of the Original Paper

For this part, the design ideas of the original paper would be provided.

First of all, VGG-19 network [12] is used as the encoder which is designed to extract the visual features from unseen style images. Then, the decoders need to be trained for transforming the output of the encoders backwards to the input images of the VGG-19 network as Figure 1. shows. Afterwards, a WCT layer is designed for processing style transferring. Finally, the combination of encoders, WCT layer and decoders would be able to achieve universal style transfer as Figure 2. and Figure 3. show. The ReLU layers shown in the Figure 3. belong to VGG-19 layer.

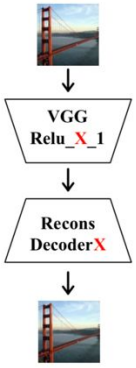


Figure 1. Reconstruction [11]

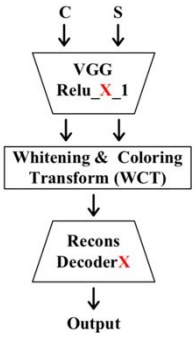


Figure 2. Single-level stylization [11]

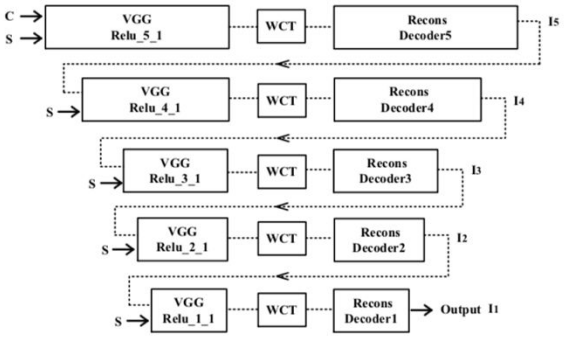


Figure 3. Multi-level stylization [11]

2.2 Key Results of the Original Paper

It is able to achieve texture synthesis by treating content image as a random noise image. Compared to other existing methods, the style-transfer results from the original paper is the only one that is able to achieve processing arbitrary style images, efficient processing and no need for pre-learning at the same time. Also, the users could control the style transfer in three different areas which are spatial control, weight and scale. The 1,200 votes show that the stylized images created by the method from original paper are uses' favorite results.

3. Methodology

3.1. Objectives and Technical Challenges

The objective of this project is to design a free-learning model which is able to transfer visual styles of arbitrary images to the other images. There are some technical challenges before achieving it. Firstly, we need to find suitable model to extract image features effectively and efficiently. Then, we need to train a decoder which is able to make the output of encoders return to the original condition. Then, a decoder criteria is needed to pick the best decoder. Finally, the last challenge is how to achieve whitening transform and coloring transform.

3.2. Problem Formulation and Design

It is time to explain how the whole system operates in detail. As the Figure 4. shows that VGG-19 network consists of 19 ReLU layers and 4 max pooling layers in total and these 19 ReLU layers could be divided into five blocks corresponding to five different visual-feature levels. To get the output of fifth-level ReLU output, two input images should go through five ReLU blocks and before the output of one block is sent to the next block, the output needs to be processed by one max pooling layer to reduce its dimensionality.

WCT is one of the major parts in the system. It transform the centered feature vector to another random vector whose covariance is the identity matrix. To inverse the whitened feature, the coloring transform is applied.[2]

For the whitening transform, the content feature vector are firstly scaled to obtain a zero mean (subtracted by a mean vector). Eigenvectors and eigenvalues of covariance matrix of the content feature vector are extracted by singular value decomposition. The vectors and values will then be used for transform the feature vector to a whitened feature.

$$f_c = f_c - m_c \quad (1)$$

$$\widehat{f_c} = E_c D_c^{-1/2} E_c^T f_c \quad (2)$$

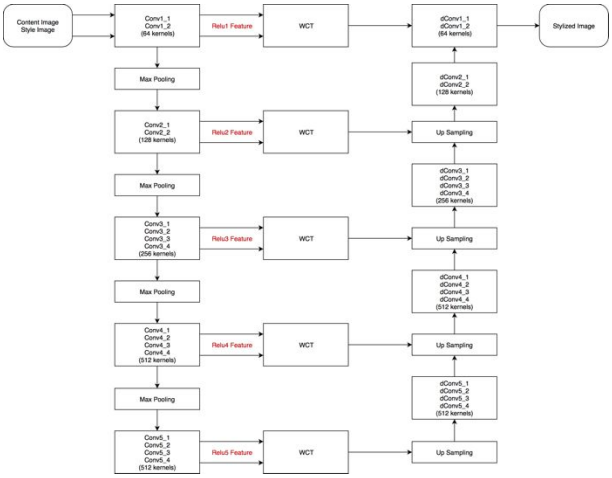


Figure 4. Style transfer flow chart

Similar with the whitening transformation, the style feature vector will subtract the mean vector at first. Then it will be further processed by the diagonal matrix with the eigenvalues of covariance matrix of style feature vector. This time it will be transferred back to a stylize content feature vector. Finally, mean vector of style is added to the feature vector recover it.

$$f_s = f_s - m_s \quad (3)$$

$$\widehat{f_{cs}} = E_s D_s^{1/2} E_s^T \widehat{f_c} \quad (4)$$

$$\widehat{f_{cs}} = \widehat{f_{cs}} + m_s \quad (5)$$

Besides, a parameter alpha serves as the style weight can be used for user to control the transfer effect.

$$\widehat{f_{cs}} = \alpha \widehat{f_{cs}} + (1 - \alpha) \widehat{f_c} \quad (6)$$

For the decoder criteria, we define a loss function. As the function (1) shows, the loss function consists of two main parts, which are feature loss [4] which is $\| \Phi(I_{out}) - \Phi(I_{in}) \|_2^2$ and reconstruction loss[8] which is $\| I_{out} - I_{in} \|_2^2$, respectively. In the loss function, I_{out} and I_{in} represent the input image of encoder and output image, respectively. Then, $\Phi(I_{out})$ and $\Phi(I_{in})$ represent the features of the input image and the output image, which are extracted by the VGG network encoder.

$$L = \| I_{out} - I_{in} \|_2^2 + \lambda \| \Phi(I_{out}) - \Phi(I_{in}) \|_2^2 \quad (7)$$

4. Implementation

To implement the style transform system, we initially train five decoders for five VGG Relu layers. WCT is applied after the encoders and before the decoders to get the final well transformed images.

4.1. Deep Learning Network

The decoders are trained in order to return the outputs from encoders to original inputs. Thus, we can assume that when it pass through WCT, the merged style image output from decoders can be distortionless to show well stylized images. Refer to figure 4 in previous page, the training procedure starts from Relu1 decoder to Relu5 decoder. Individual decoder is trained separately and decode the output from WCT to give a stylized image.

To train a decoder, the algorithm is shown as the figure 5. When images are input to the encoder, feature vectors are generated. Then we feed them into the decoder and calculate the loss using original inputs (I_{in}), feature vectors from encoders ($\Phi(I_{in})$), reproduced images (I_{out}) and the feature vectors from the reproduced images ($\Phi(I_{out})$). Calculated loss will be applied for back propagation for adjusting the parameters in decoders.

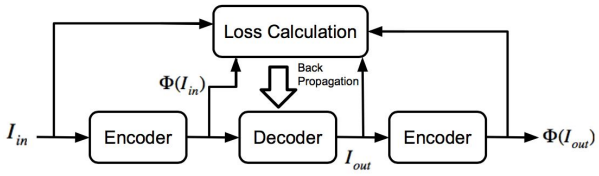


Figure 5. Flowchart for training decoders

4.2. Software Design

The system is trained on the Microsoft COCO dataset. It contains over 80,000 images. The batch size of each training is 8 images and we trained Relu5 decoder for 40,000 steps and other layers for 20,000 steps.

After the decoders are prepared, WCT will be applied for computation for the stylized images. According to its definition, we can easily program it in Python. The pseudo code shown below can give a clear indication on how to implement WCT.

Pseudo code for WCT:

Input: $\widehat{f_c}$*: content feature vectors,* $\widehat{f_s}$*: style feature vectors*

output: $\widehat{f_{cs}}$*: transformed feature vectors*

$\widehat{f_c} = \widehat{f_c} - mean(\widehat{f_c})$
Do SVD on Covariance($\widehat{f_c}$*).*
Get E_c: matrix of eigenvectors, D_c:diagonal matrix with eigenvalues.

$\widehat{f_c} = E_c\, D_c^{-1/2}\, E_c^T\, f_c$
Do the same for $\widehat{f_s}$ *to get* E_s , D_s

$\widehat{f_{cs}} = E_s\, D_s^{1/2}\, E_s^T\, \widehat{f_c}$
$\widehat{f_{cs}} = \widehat{f_{cs}} + mean(\widehat{f_s})$
$\widehat{f_{cs}} = alpha * \widehat{f_{cs}} + (1\text{-}alpha) * (\widehat{f_c} + mean(\widehat{f_c}))$

As the Figure 6. shows that the whole system consists of four main parts, which are encoder (pre-trained VGG-19 network), WCT layer, decoder (reversed VGG-19 network) and decoder trainer. The input of VGG-19 network would be one style image and one content image, and the network could exact the features from two images. Then, WCT layer would execute a whitening transform and a coloring transform in turn and as the result it would output transformed features. Afterwards, the output of WCT would be sent to decoder and the decoder would process these transformed features to obtain stylized images.

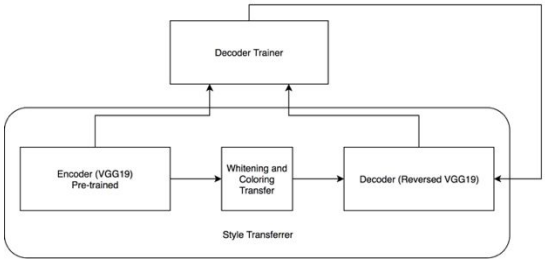


Figure 6. System block diagram

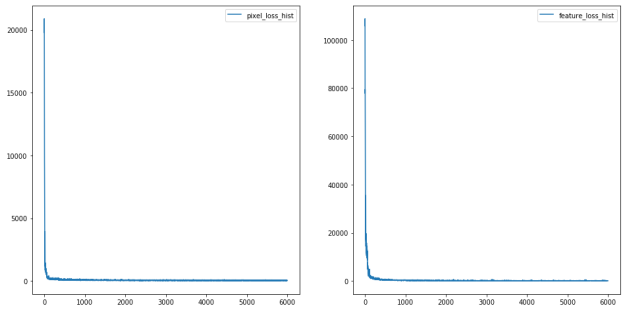
5. Results

5.1. Project Results

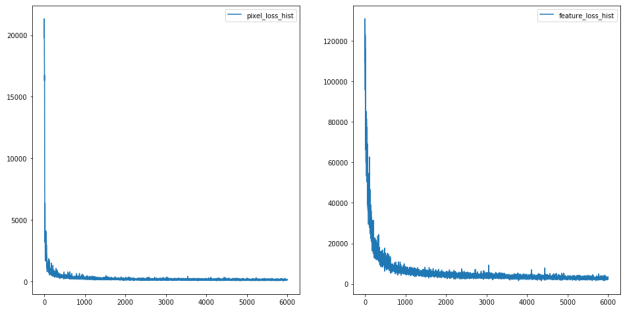
Loss values are recorded when we train the five separate decoders. Since in the training process, the decoders are always adjusted in a way of minimizing loss, the performance of the decoder can be observed from the plots for loss over steps.

Figure 7 has shown five pixel loss and feature loss for five decoders respectively. Obviously, the loss drops quickly from extreme high value down to a low value near zero within 1000 steps in Relu1 decoder. When the neural network gets more complex and has more

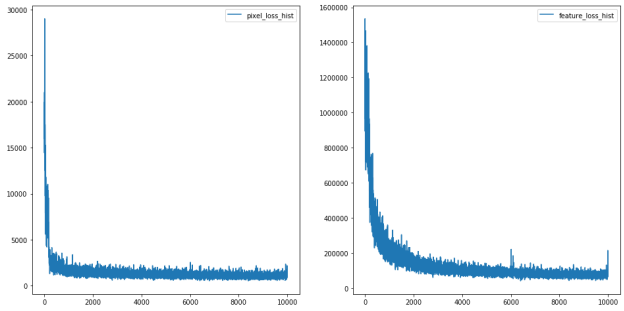
parameters, the number of small oscillations increase. The slope of plots in (b) is a little bit smoother than the Relu1 decoder loss curve, even though it still converges very fast. Same thing happened for other decoders. The loss values oscillate more frequently. In other word, their loss values keep fluctuate at some relatively low levels. In addition, their convergence speeds become slower. Loss in Relu3 converges in the first 2000 steps. Relu4 has its feature loss value to drop to some certain level within 7500 steps. As for Relu5, the pixel loss will drop within the first 5000 steps. However, because the scale of the feature is only 14*14, their feature loss is alway fluctuating at a low level but never converge.



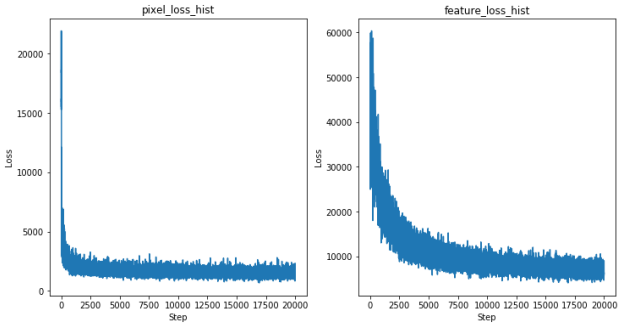
(a) Loss plots for Relu1 decoder



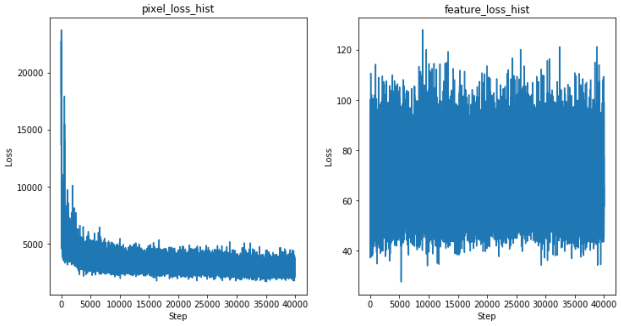
(b) Loss plots for Relu2 decoder



(c) Loss plots for Relu3 decoder



(d) Loss plots for Relu4 decoder



(e) Loss plots for Relu5 decoder

Figure 7. Loss for five decoders

The fact is that, the simpler the neural network is, the easier to train the decoder. As for Relu5, if a converged plot in feature loss is required, more steps are needed to train it which may cost days to complete the training procedure.

Now we combine five decoders, WCT module and VGG encoders to a transform system. Here we separately test the transform effect of each Relu layer in the system. The fact is shown in figure 8. We can see that when the layer becomes deeper, the merged style gains higher level information. However, we can see that for the Relu5 layer, the decoder cannot correctly restore the original image. This is because we did not train the Relu5 decoder for enough steps. The Relu5 decoder is too complicated to train, so that we cannot improve it to obtain a satisfiable performance in limited time.

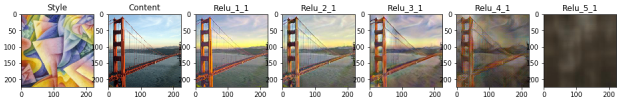


Figure 8. Images transferred through single layer

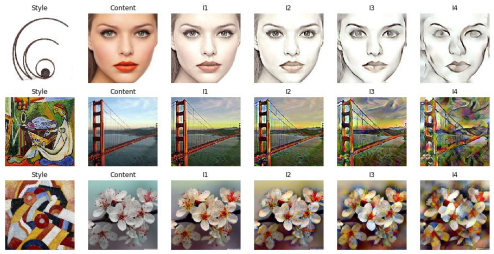
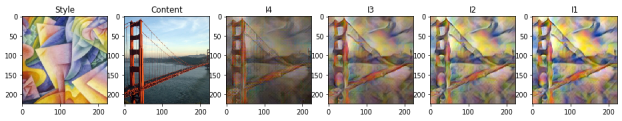


Figure 9. Images transferred through system

With all the decoders we trained, we can construct a multi-level stylization system. Here, we abandon the Relu5 decoder because of its bad performance. Only four decoders are applied to the system. In figure 9, the effects of the multi-level stylization system from one layer to four layers are displayed. It is obvious that the multi-level stylization system performs well not only on a specified content or style, but its utility is universal. In other aspect, when the neural network becomes deeper, the output will have a higher level of information and the style is more distinct.

5.2. Comparison of Results

In this section , we would like to compare the resulting images generated from the system we build with the original paper results. In this regard, the style and content images which are used in original paper are input to the system for testing. The results are shown in graph (a) in figure 10. It is pretty similar with the results of the original paper. Especially the final outputs (image I_1) of the paper and our system look just alike at the first glance. The details do differ in some way. Our results are more coarse than the original paper results. This may mainly due to the fact that good decoders require more steps to train, and the original paper may trained the decoders with numerous steps until well-trained decoders are obtained. The running time for achieving visual features transfer is about 3.89 seconds. It is quite similar compared to the results of the original paper.



(a) Images transferred through system

(a) I_5 (b) I_4 (c) I_1

(b) Transformed results from original paper

Figure 10. results of system and original paper

5.3. Discussion of Insights Gained

We completely reproduce the system described in the original paper. The system architecture is completely the same as the flow chart, and the results are very similar. But we also faced some difficulties training the model.

First, the training of the decoders is very time-consuming and resource-consuming. The MSCOCO dataset we used contains more than 80,000 images. It is impossible to read all the images into the memory at once and feed them into the network. The only way to do so is to read images in sequence. In our implementation, the system will read about 1000 images at a time, training the network with them and then read another 1000 images, which increase the complexity of the logic of training function. The convergence time of training each decoder also increases dramatically with the CNN structure becomes deeper. For example, it takes only 500 steps to make the decoder of Relu1_1 layer converge. However, 40,000 steps, which cost about 15 hours to train, is still far from enough to train the decoder of Relu5_1 layer to restore the original image precisely.

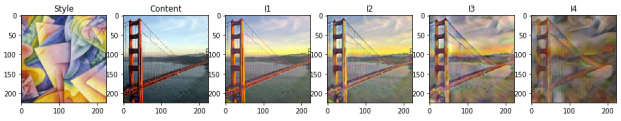


Figure 11. Fine-to-coarse

Second, we find that each layer in the system transfers the feature of the input images in different level. For example, the Relu1_1 layer usually only adds the color information of style image to the content image. While Relu4_1 layer are not good at containing color information. It is usually much darker than Relu1 output. However, the Relu4_1 layer is good at transfer line information of the style. This makes the multi-level stylization system performs much better than the single layer. Here we also did a comparison between the coarse-to-fine stylization and a fine-to-coarse stylization, which is a reverse structure that image transfer from Relu1 layer to Relu4 layer. The original one, which is coarse-to-fine stylization, contains the style information from all the transfer layer while the

latter one can only contain the highest level. This is because the deeper transfer layer has limited ability in containing the low level features. Therefore, the low level style features can be hardly merged into the content feature matrix in deeper layer.

Third, we observe that, while the weight lambda in the loss function when training the decoder is set to 1. The actually pixel loss and the feature loss can be different in the order of magnitudes. Therefore, total loss of the loss function usually only reflects either pixel loss or feature loss. In the first four level, the decoder usually have larger feature loss, while for layer 5, the feature loss is extremely small. Modifying the weight of the feature loss in the loss function may do better job in training the decoders.

The style weight in WCT is also a very important parameter in affecting the magnitude of the stylization. The larger it is, the stronger content image is stylized. If we set this weight too large, the content might be seriously distorted that we cannot even recognized it. In our system, we find that a style weight of 0.6 is the best for the stylization.

6. Conclusion

This project implements a multi-level stylization system which is composed by VGG encoders, WCT module and decoders. This is a complete reproduction of paper *Universal Style Transfer via Feature Transform*. Besides the WCT module we programed, five decoders are trained for the system. And the simpler the neural network is, the easier to train the decoder. Furthemore, the multi-level stylization system utility is universal and when the neural network becomes deeper, the output will have a higher level of information and the style is more distinct.

By comparing the results generated from our system and the results displayed in paper, we can find that our system does pretty well on style transform with the first four well-trained Relu decoders. Our deficiency is that we do not have enough time to train a better Relu5 decoder. The Relu5 decoder requires numerous steps to train, which may cost days to operate. Furthemore, The overall performance can be improved as well if the decoders are trained with more steps.

7. Acknowledgement

We would like to express our thanks of gratitude for Prof. Zoran Kostic who always glad to help us with valuable advices which benefits us a lot when work on the project.

8. References

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

[1] Bitbucket Link:
https://bitbucket.org/ecbm4040/2018_assignment2_tg2663/src/master/
https://bitbucket.org/ecbm4040/2018_assignment2_hy2557/src/master/
https://bitbucket.org/ecbm4040/2018_assignment2_yz3397/src/master/

[2] Maliha Hossain, Whitening and Coloring Transformations for Multivariate Gaussian Data, A slecture for ECE 662. https://www.projectrhea.org/rhea/images/1/15/Slecture_ECE662_Whitening_and_Coloring_Transforms_S14_MH.pdf

[3] D. Chen, L. Yuan, J. Liao, N. Yu, and G. Hua. Stylebank: An explicit representation for neural image style transfer. In *CVPR*, 2017.

[4] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In *NIPS*, 2016.

[5] V. Dumoulin, J. Shlens, and M. Kudlur. A learned representation for artistic style. In *ICLR*, 2017.

[6] L. A. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *NIPS*, 2015.

[7] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016.

[8] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In*ECCV*, 2016.

[9] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. Diversified texture synthesis with feed-forward networks. In *CVPR*, 2017.

[10] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, 2016.

[11] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yangy. Universal Style Transfer via Feature Transforms. *arXiv preprint arXiv: 1705.08086v2*, 2017.

[12] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition.*arXiv preprint arXiv:1409.1556*, 2014.

9. Appendix

9.1 Individual student contributions in fractions - table

	tg2663	hy2557	yz3397
Last Name	Guan	Yan	Zhang
Fraction of (useful) total contribution	1/3	1/3	1/3
What I did 1	WCT	Decoder	Vgg
What I did 2	Report	Report	Report