Style Transfer Image Background

Weisheng Wang
Department of Computer Science
University of California, Irvine
weishew@uci.edu
72091392

Sirui Hu
Department of Computer Science
University of California, Irvine
siruih1@uci.edu
48235202

Kanglan Tang
Department of Computer Science
University of California, Irvine
kanglant@uci.edu
26497661

Longxiang Dai
Department of Computer Science
University of California, Irvine
longxiad@uci.edu
70961656

Qian Zhao
Department of Computer Science
University of California, Irvine
qzhao9@uci.edu
67157879

Abstract

In recent years, there is enormous progress in the domain of image style transfer, such as the invention of Gram matrix evaluation metrics and the very first neural network model by Gatys *et al*. In our project, we aim to apply style transfer to the background of an input photo in real-time and leave the main subject(s) (e.g. people) in the photo unchanged. To enhance the visual quality of the generated image, we blend the edge between the subject(s) and the transferred background. We build two kinds of style transfer networks upon the real-time style transfer network by Johnson *et al* and the MSG-Net by Zhang *et al*. We improve the original models by making several modifications such as VGG downsampling, multiple CoMatch, and blending loss. Compared to original networks, our networks retain the visual qualitative results while the original main subject(s) appears naturally in the generated image. Our networks also outperform the original networks in various aspects.

1 Introduction

Many people like making their photos more creative and less dull. In our project, we aim to allow people(or other subjects) to appear in an artistic and magical background in each photo. We accomplish the task of style transferring the background of the input image by building two kinds of real-time style transfer networks, one for a single style and another for multiple styles.

Here is our application workflow(shown in Figure 1): first, use a pre-trained object detection model, Mask-RCNN [4], to detect specified objects in the image and to generate the mask for the objects; then, style transfer the masked image with our style transfer networks; finally, put the original image portion of the subjects back to the generated image and blend the edge.

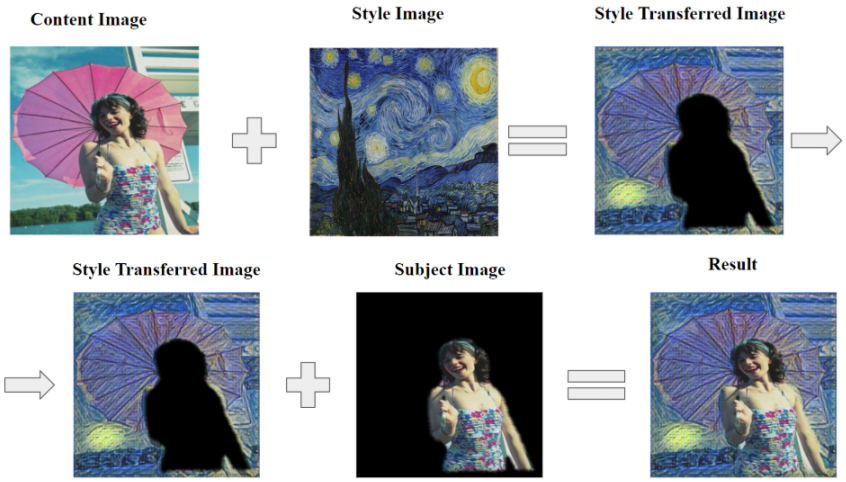


Figure 1: An example of our application workflow.

This paper is organized as follows. Section 1.1 reviews related works as well as ideas for this application. Section 2 introduces our modifications on the original single-style style transfer network(Section 2.1) and the original multi-style style transfer network(Section 2.2), and reviews loss functions(Section 2.3). Then, we discuss the dataset, preprocessing approaches(Section 3.1), and experiments with the single-style networks(Section 3.2) and the multi-style networks(Section 3.3). We further analyze different blending approaches and evaluate our blending loss(Section 3.4). Next, we compare the networks of various architectures(Section 3.5). Finally, we discuss possible future improvements and conclude the paper(Section 4).

1.1 Related work

1.1.1 A neural algorithm of artistic style model

Gatys *et al* [1,2] combine the content of the input image and the style of the style image to perform the style transfer task. The model first uses an arbitrary image to start with and modifies that image based on the content loss [10] and style loss [2] functions. Although his contribution is significant in the style transfer domain, his model is computationally expensive and fails to satisfy the need for real-time generation.

1.1.2 Perceptual loss for real-time style transfer model

The image transformation network and loss network are two critical methods in Johnson *et al* [6]. After the network is trained for one style, it can generate a transferred image within seconds. It first goes through several convolution layers for downsampling and then a couple of residual blocks. Next, it will go through three convolution layers for upsampling. Finally, the network will calculate style loss, content loss, and total variation loss using the pretrained VGG16 [12].

1.1.3 Multi-Style generative network for real-time transfer

Zhang *et al* [15] utilizes CoMatch and bottleneck architecture to build multi-style style transfer networks. The CoMatch layer enables the network to perform real-time multi-style generation using a feed-forward network; the bottleneck structure helps to compress feature representations to improve the performance of the network during training(details about CoMatch layer and bottleneck structure designs are in appendix). Moreover, Zhang points out that the new upsample convolution approach reduces the checkerboard artifacts.

2 Method

In this section, we will introduce our modifications to the original single-style style transfer network by Johnson *et al* and the original multi-style style transfer network by Zhang *et al*. We will also introduce common metrics to measure the performance of style transfer networks. We

build our networks on top of the basic architecture of the original networks(exact architectural details are in the appendix).

2.1 Single-style style transfer network

2.1.1 VGG Downsampling

In the original network by Johnson *et al* [6], the author first uses one 9x9 convolution layer and two stride-2 3x3 convolution layers to downsample the image y from (256, 256, 3) to (64, 64, 128). (Activation size: height, width, channels) In our networks, we use the pre-trained VGG19 network to replace the original downsample structure. We use VGG19 for downsampling because VGG19 is capable of extracting latent feature maps of the input image. Since one of the main purposes of downsampling is to extract the latent feature representations of the image, we hypothesize that using a pre-trained network for downsampling will increase the speed of convergence for our networks. In the experiment section, we will present how switching to use VGG for downsampling improves the performance of our networks. We also find a reference paper [8] that uses the VGG network to extract the downsampled feature map which further justifies the rationale of our approach.

2.1.2 Bottleneck Upsampling

Inspired by Zhang *et al* [15], we extend the bottleneck architecture to an upsampling residual block (as shown in Figure 13 in appendix). In our networks, transposed convolution layers are replaced by upsampling residual blocks. As suggested in the paper, the architecture reduces computational complexity without losing versatility by preserving a larger number of channels. Compared to the standard convolutional neural networks which use fractional strides that results in checkerboard artifacts, the upsample residual block helps to reduce the problem and improves the quality of the generated image.

2.1.3 Instance Normalization

In the original network by Johnson *et al* [6], all non-residual convolutional layers are normalized by batch normalization. While using batch normalization is common, Ulyanov [13] proposes the instance normalization approach, as shown in Figure 11, which brings a significant qualitative improvement in the generated images of stylization networks. Since instance normalization improves the visual quality of generated images, in our single-style and multi-style networks, we adopt this normalization approach. Calculation details of instance normalization is in the appendix.

2.2 Multi-style style transfer network

2.2.1 VGG Downsampling

Learned from the effectiveness of using VGG19 network to downsample in our experiment in the single-style network, we replace the siamese network in Zhang's model [15] with the pretrained VGG19 network. As shown in Figure 18 in appendix, we extract the input image using VGG19 layer block3_conv3 and then pass it through a convolution layer to increase the filter size to 512 to make the feature map's size compatible for the CoMatch layer proposed by Zhang *et al*.

2.2.2 Multiple Comatch

Inspired by Li *et al* [7], we combine the idea of residual block with multiple comatch layers hoping that the model can learn more features from both style and content images while retaining the old performance. The siamese network is prolonged after the CoMatch layer by replacing the original six bottleneck layers in a single network with three bottleneck layers in the siamese network. Additional three CoMatch layers are inserted after each bottleneck layer. The result of each CoMatch layer is added onto the next CoMatch layer to form a residual block layer. The architectural details can be found in Figure 19 in the appendix.

2.2.3 Multiple Comatch while downsampling

Motivated by Multiple CoMatch layers and hoping to learn more features from style and content images, we implement some additional CoMatch layers during downsampling. We break the downsampling model into 3 parts and insert 3 more CoMatch layers after the convolution layer and the bottleneck layer. One key difference of CoMatch layers on the downsampling network from those on the residual block is that the size of each CoMatch layer changes by sharing weights with

content network and style network during downsampling while the size of those on the residual block remains the same. The architectural details can be found in Figure 19 in the appendix.

2.2.4 UNet

We specifically build the UNet version of the multi-style network after adding the blending loss function because UNet is well-known in predicting segmentation maps, and the task of retaining the subject can be classified as segmentation. The mask is applied to the content image beforehand and the network needs to learn the location of the mask, so we pick the content image side of the siamese network to do UNet. Figure 20 in appendix shows the new architecture with UNet. In the expansion/upsampling path, we concatenate the feature maps with the corresponding feature maps of the content image in the contracting/downsampling path of the siamese network.

2.3 Loss functions

For our models, we use two perceptual loss functions, feature reconstruction loss and style reconstruction loss, to measure the style differences and semantic feature differences between images. Both loss functions make use of a pretrained network φ as a loss network to perform the measurement. In our models, φ is the 19-layer VGG network trained on the ImageNet dataset[11]. Moreover, to output visually-pleasing images, we use total variation loss, and devise our own blending loss to blend the edge between style-transferred background and original subject.

2.3.1 Feature Reconstruction Loss

Johnson *et al* [6] uses feature reconstruction loss(content loss) function to measure the semantic feature representation differences between the output image and the target image by the pretrained loss network φ. Johnson defines that: let $\varphi_j(x)$ be the activation results of the jth layer of the network φ when the network processes the image x; if j is a convolutional layer then $\varphi_j(x)$ will be a feature map of shape $C_j \times H_j \times W_j$. The feature reconstruction loss is the normalized squared Euclidean distance between the feature representations of the output image and the target image:

$$l_{feat}^{\varphi,j}(\hat{y},\mathrm{y}) = \frac{1}{C_j H_j W_j} \|\varphi_i(\hat{y}) - \varphi_i(\hat{y})\|_2^2 \qquad (1)$$

2.3.2 Style Reconstruction Loss

Gatys *et al* [1, 2] proposes style reconstruction loss(style loss) to measure the style differences such as color, texture, common pattern and other stylistic differences. Gatys defines that: let $\varphi_j(x)$ be the activation results of the j[th] layer of the network φ for input image x which is a feature map of shape $C_j \times H_j \times W_j$. Define Gram matrix $G_j^{\varphi}(x)$ to be the $C_j \times C_j$ matrix which is the result of the following formula:

$$G_j^{\varphi}(\mathrm{x})_{c,c'} = \frac{1}{C_j H_j W_j} \sum_{h=1}^{H_j} \sum_{w=1}^{W_j} \varphi_j(\mathrm{x})_{h,w,c} \varphi_j(\mathrm{x})_{h,w,c'} \quad (2)$$

The style reconstruction loss is then the squared Frobenius norm of the difference between the Gram matrices of the output image and the target image.

$$l_{style}^{\varphi,j}(\hat{y},\mathrm{y}) = \left\|G_j^{\varphi}(\hat{y}) - G_j^{\varphi}(\mathrm{y})\right\|_F^2 \qquad (3)$$

2.3.3 Total Variation Regularization

Mahendran *et al* [10] uses total variation regularization(TV) to encourage neural networks to generate smooth images:

$$R_{V\beta}(\mathrm{x}) = \sum_{i,j} ((x_{i,j+1} - x_{ij})^2 + (x_{i+1,j} - x_{ij})^2)^{\beta/2} \quad (4)$$

where β = 1.

2.3.4 Blending Loss

We implement blending loss $l_{blending}$ in our models to keep the subject of interest unchanged by penalizing the model for changing the value of the subject in the masked area.

The blending loss is defined as:

$$l_{blending} = \frac{\alpha \times \sum_{i=0}^{h-1} \sum_{j=0}^{w-1} \left[M_{ij} \times \left(\left(\varphi(x_c, x_s) \right)_{ij} - (x_c)_{ij} \right)^2 \right]}{S} \quad (5)$$

$$S = \sum_{i=0}^{h-1} \sum_{j=0}^{w-1} I\left(M_{ij} == 1\right) + 1 \quad (6)$$

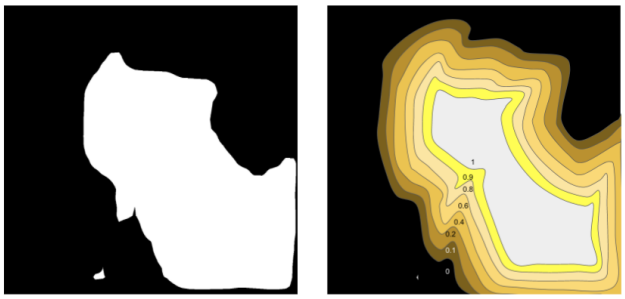
It is simply the Euclidean distance between the output image and the content image. We need to specify the blending loss weight which is α. M is either a complete mask or a percentage mask. If M is a complete mask, we penalize the model for changing all the places within the subject mask. If M is a percentage mask, we penalize more draconically if the model changes the core of the subject mask while allowing for relatively small changes to the periphery of the subject. The variable S represents the size of the subject area which is the number of the subject's and the edges' pixels. In case the mask is empty, we regularize it by adding 1. The function I is the indicator function.

3 Experiment

3.1 Dataset

The dataset to train all the models is the Microsoft 2014 COCO training dataset [9]. The dataset has more than 80k images which are in color or in black and white. We first resize all the images to (256, 256, 3). For black and white images, we stacked the same array to the RGB channels so that the channel size increases from 1 to 3.

To prepare the dataset for our networks implemented with blending loss, we use Mask RCNN [4,5] to generate a complete mask for each resized image in the COCO dataset. A complete mask has the size of (256, 256, 1) and stores only 0 and 1 to indicate whether the pixel is classified as background or subject. In our experiments, we only generate the mask for humans and cats. Since we want the effect of blending on the edges of the subjects, we also generate a percentage mask based on the complete mask to store the percentage of the background to keep along the edges. As shown in Figure 2, the closer the layer is to the core of the subject, the lower the percentage of the background is.



Complete Mask Percentage Mask

Figure 2: The percentages represent how much the subject needs to be kept. As shown in the figure, there are 8 layers. From the edge in the complete mask, we expand four layers both inside and

outside the edge. The width of each layer is one pixel.

3.2 Single-style

3.2.1 VGG downsampling

In the experiment, we train our networks on two different styles: Picasso's Dora Maar and Van Gogh's starry night. When all networks assign the same weights for various losses, we find that using VGG19 for downsampling can reduce a considerable amount of content loss and total variation loss. In other words, using VGG19 for downsampling is more efficient in extracting the latent feature representations of the input image so that the style-transfer networks are more capable of reconstructing the features. However, such capability does not appear significantly in extracting stylistic features.

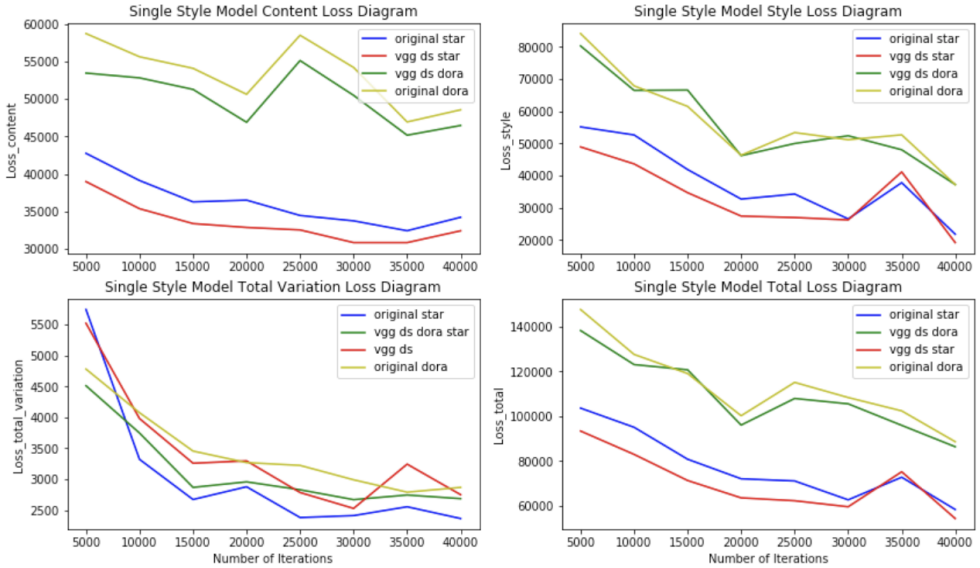
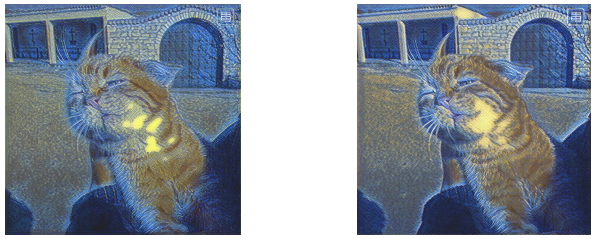


Figure 3: When the iteration number is the same, the networks using VGG19 for downsampling always have a lower content loss and a lower total variation loss. In the style loss diagram for the starry night style, most of the time, the network with VGG19 downsampling has a lower style loss. However, for the Dora style, we cannot observe such differences.

While the network architect has changed, the networks still consistently generate visually pleasing style-transferred images (Figure 4). This phenomenon suggests that replacing the original downsampling layers by the VGG19 network is effective and practical.



Original network Network with VGG19 downsampling

Figure 4: The left image is the generated image of the original network. The right image is the generated image of the network using VGG19 for downsampling. Both networks train in starry night style.

In addition, using VGG19 for downsampling reduces 100,608 parameters for training which is

about 5.9% of the total number of parameters in the original architecture.

3.2.2 Bottleneck Upsampling

In the experiment, we find that using bottleneck for upsampling reduces the total variation loss (Figure 5) and makes the networks generate smoother output images(Figure 6).

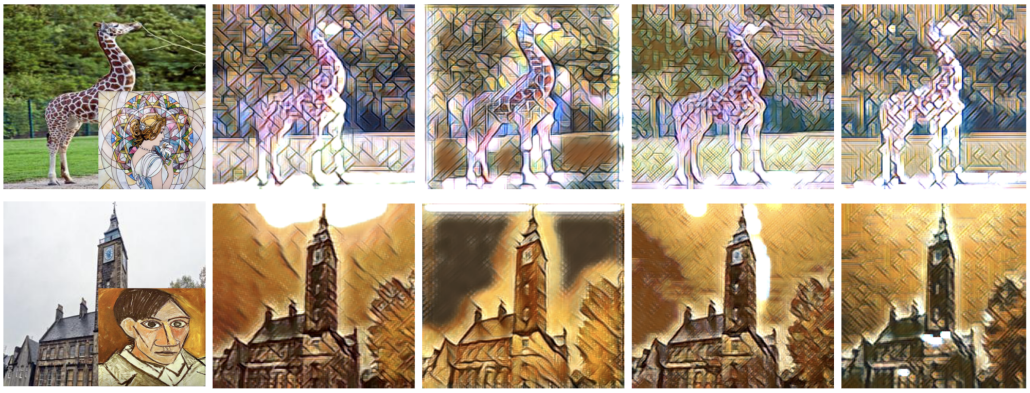
Figure 5: This is the total variation loss diagram of the networks. As we can observe, most of the time, networks using upsampling bottlenecks have a smaller total variation loss.

Original Bottleneck for upsampling

Figure 6: Left: Image generated by Johnson's network. Right: Image generated by upsampling bottleneck. We can observe that using bottleneck, the generated image is much smoother.

3.3 Multi-style

We experiment with different multi-style style-transfer models, including the original MSG-Net by Zhang *et al* [15] and models with VGG downsampling, multiple CoMatch, and multiple CoMatch during downsampling. We hope the VGG downsampling can improve the efficiency of the network training process and the network is able to learn more features of the style image and content image through multiple CoMatch layers.



(a) input (b) MSG-Net (c) VGG Downsampling (d) Multiple CoMatch (e) Multiple CoMatch during Downsampling

Figure 7: Examples of style-transferred images by different models.

Possibly because it is faster to learn a small customized network for the style image that is different from the normal classification, using the pre-trained VGG19 network to downsample content image and style image does not speed up the learning process. However, we discover the giraffe image(c) in Figure 7 preserves more color of the style image than other images. And both the building and the giraffe are more imposing because of a bright outline around the subject. It proves that the VGG19 network is more capable of extracting the latent feature representations of the input images than the original siamese network.

As shown in Figure 7 we can observe that using multiple CoMatch layers help to gain more features of the input images than the original model. We observe that the outline of the object in the image is more obvious and the background contains more delicate style patterns. Then, by adding additional CoMatch layers during downsampling, the model shows more capability in dealing with a large area with the same color. As we can see in the building image (c), the colors distribute more evenly for the sky and, visually, it retains more contents than other images.

3.4 Blending

We come up with 5 alternatives to retain the subjects in the style transferred image: copy/paste, partial average and weighted average.

Copy/paste: The simplest way is to copy and paste the area within the mask from the content image onto the transferred image.

$$\hat{y}_{ij} = I(M_{ij}! = 1) \times (\varphi(x_c, x_s)_{ij}) + I(M_{ij} == 1) \times (x_c)_{ij} \qquad (7)$$

Partial Average: Assuming blending loss using percentage masks can blend the edges of the result, we add the reduced edge values of the content image onto the style transferred image.

$$\hat{y}_{ij} = (\varphi(x_c, x_s)_{ij}) + M_{ij} \times (x_c)_{ij} \qquad (8)$$

Weighted Average: The new image would be the weighted average of the transferred image and the content image using the weight given by a percentage mask.

$$\hat{y}_{ij} = (1 - M_{ij}) \times (\varphi(x_c, x_s)_{ij}) + M_{ij} \times (x_c)_{ij} \qquad (9)$$

However, if we do not add the blending loss function in the model, the contrast of colors between nearby layers is evident. So, we add the blending loss to our model to introduce the randomness and shorten the time of the process. We also use the percentage mask in the blending loss and

preprocessing. As we can see from Figure 8, this approach generates the best result.

The whole process of this approach(as shown in Figure 9) is: (1) contract the mask to contain only the core of the subject which means the values of that area can only be 1 in the percentage mask, (2) make the masked area black in the content image before the content image was passed into the model, (3) generate a weighted average result of the image from the style transferred image and the content image.

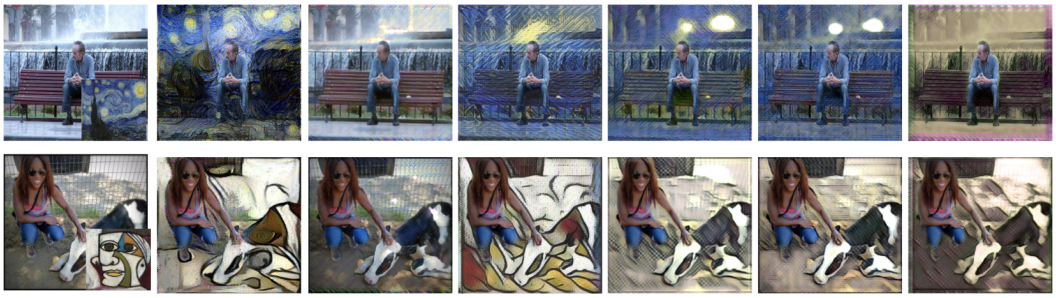
(a) Copy/Paste (b) Partial Average (c) Weighted Average

Figure 8: The "copy/paste" method has a black outline of the subject. The "partial average" result has an obvious white outline of the subject that half of the man's face is obscured. The "weighted average" result has the overall best effect.

(a) Content Image (b) Complete Mask (c) Percentage Mask (d) Content Image w/ Percentage Mask

Figure 9: From a to d is a sequence of preprocessing the data before the image was passed into the model.

3.5 Final Result Comparison



(a) input (b) Gatys complete mask (c) MSG-Net complete mask (d) Single-style w/ VGG downsampling (masked) percentage mask (e) MSG-Net (masked) percentage mask (f) Multiple CoMatch (masked) percentage mask (g) UNet (masked) percentage mask

Figure 10: Comparison of models with different architectures, data preprocessing, and blending loss

function. The label *masked* indicates whether a mask is applied onto the image in preprocessing. The label *complete/percentage mask* indicates which type of mask is used in data preprocessing and blending loss function. Model d is trained 2 epochs with a batch size of 4 on the 80,500 images. Models e to g are trained with 1 epoch with the same batch size and dataset.

Adding the blending loss with a complete mask into Gatys' model [1,2] has a quite pleasing result. The background is style transferred fully while the subjects of interest are not changed at all. However, the edges of the subject are still observable so the problem of blending is not resolved because the model uses pixel loss.

So we add the blending loss only to the MSG-Net model. According to the result, it is too heavy for the model to take all the tasks at once: segmentation, style transfer, and blending. As we can see from image(c) in Figure 10, only a limited style transfer effect on the result can be observed.

Then we apply the mask to the content image in preprocessing to lower the difficulty of learning because subjects in black are easier for networks to distinguish from background than subjects in color. The single-style model using VGG19 downsampling and bottleneck upsampling has the best result overall. The background is style transferred completely while retaining some basic outlines of the objects, such as the bench(in images d). The sitting man is also preserved with nice shape and soft edges.

The results of multi-style models (images e-g) do not have very high quality. The subject is retained and does blend with the background but the background is not transferred as obvious. It is possible that the heavier task makes the model learn slower than before.

The UNet version is a little bit faster than MSG-Net version in style transferring the image which is more desirable in real-time transfer. Though the result of the woman and cow is acceptable, the UNet is not good at learning complex styles such as the Starry Night by Van Gogh.

4 Conclusion

Based on the work of Johnson *et al* [6] and Zhang *et al* [15], we conduct multiple experiments on the fast-style transfer network and multi-style transfer network. To improve the efficiency and performance of the original networks, we make several modifications to the architectural designs, including VGG downsampling, bottleneck upsampling, multiple CoMatch layers, etc. In addition, to keep the selected subject(s) in an image the same and to improve the visual quality of the generated images, we try out various methods of masking and edge blending and introduce a blending loss function for the training process. Our models have achieved good image quality of the style transformation, and the blending loss technique has produced smooth edges between the subject(s) and the transferred background, which enhances the visual quality of the background style transformation while leaving the subjects unchanged.

There is still room for improvements. Because of the hardware limitation, we did not get to train our models long enough that the performance of some multi-style models are not as satisfying. If we train our models longer, the results in Section 3.5 might be improved.

Another possible way is to split the task into two models. One model only does style transfer. Another model segments the subject of interest and blends the edges at one time. This may make the task more flexible to switch between different models of style transfer without the need to train a new model with the blending loss.

Additionally, since VGG19 has already brought a considerable improvement to our networks, we believe that there exist other downsampling approaches to improve the performance of the networks. For instance, we can pretrain an autoencoder to do such a task.

5 Acknowledgments

This work is done using Tensorflow 1.4 and 2, FloydHub GPU, local GeForce GTX 1070 GPU, and Google Collaboratory GPU. Thanks to the support from Prof. Pierre Baldi. We still remember that during the Q&A section of our presentation, professor Baldi asked us why we chose this topic. At a first thought, we thought that we chose the topic because many of us have experience with image processing. However, looking back to what our team had been through, either the frustrations during brain-strorming and coding the networks or the joyment when we observed the performance improvement to the networks, we found that it was really the passion of exploring and manipulating and implementing various neural network architectures that supported us till the end. Also thanks to TA Julian Collado and TA Ruihan Yang for patiently answering our questions and making useful suggestions for our network architectural designs during their office hours.

6 References

[1]Gatys, L.A., Ecker, A.S., Bethge, M.: A neural algorithm of artistic style. arXiv preprint arXiv:1508.06576, 2015.

[2]Gatys, L.A., Ecker, A.S., Bethge, M.: Texture synthesis using convolutional neural networks. In: Advances in Neural Information Processing Systems 28, 2015.

[3]Gross, S., Wilber, M.: Training and investigating residual nets. http://torch.ch/blog/2016/02/04/resnets.html, 2016

[4]He, Kaiming, et al. "Mask r-cnn." Proceedings of the IEEE international conference on computer vision. 2017.

[5]He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[6]Johnson, Justin, Alexandre Alahi, and Li Fei-Fei. "Perceptual losses for real-time style transfer and super-resolution." *European conference on computer vision*. Springer, Cham, 2016.

[7]Li, Bo, et al. "Siamrpn++: Evolution of siamese visual tracking with very deep networks." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019.

[8]Li, Chuan, and Michael Wand. "Precomputed real-time texture synthesis with markovian generative adversarial networks." *European conference on computer vision*. Springer, Cham, 2016.

[9]Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Doll´ar, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Computer Vision– ECCV 2014. Springer 740–755, 2014.

[10]Mahendran, A., Vedaldi, A.: Understanding deep image representations by inverting them. In: Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). (2015)

[11]Russakovsky, O., Deng, J., Su, H., et al. "ImageNet Large Scale Visual Recognition Challenge." *International Journal of Computer Vision (IJCV)*. Vol 115, Issue 3, pp. 211–252, 2015.

[12]Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556, 2014.

[13]Ulyanov, Dmitry, Andrea Vedaldi, and Victor Lempitsky. "Instance normalization: The missing ingredient for fast stylization." *arXiv preprint arXiv:1607.08022*, 2016.

[14]Wu, Yuxin, and Kaiming He. "Group normalization." Proceedings of the European Conference on Computer Vision (ECCV). 2018.

[15]Zhang, Hang, and Kristin Dana. "Multi-style generative network for real-time transfer." *arXiv preprint arXiv:1703.06953*, 2017.

7 Appendix

7.1 Instance normalization

Instance normalization is computed as follows:

$$y_{tijk} = \frac{x_{tijk} - \mu_{ti}}{\sqrt{\sigma_{ti}^2 + \epsilon}}, \mu_{ti} = \frac{1}{HW}\sum_{l=1}^{W}\sum_{m=1}^{H} x_{tilm}, \sigma_{ti}^2 = \frac{1}{HW}\sum_{l=1}^{W}\sum_{m=1}^{H}(x_{tilm} - m\mu_{ti})^2 \quad (10)$$

Below, we use the figure from Wu *et al* [14] to demonstrate the difference between batch norm and instance norm visually:

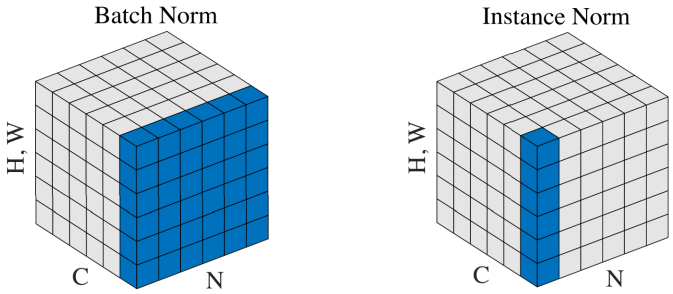


Figure 11: each cube represents a feature map tensor; N is the batch size; C is the number of channels; (H, W) is the height and the width. The blue pixels are normalized by the same mean and variance, which are obtained by the values of these pixels.

7.2 Bottleneck

Gross *et al* [3] proposed the bottleneck architecture, an alternative to residual block, which builds deeper architectures. Figure 12 shows that the unit consists of three stacked operations: 1x1, 3x3, and 1x1 convolutions. The two 1x1 layers are designed for reducing and restoring dimensions. This leaves the middle 3x3 layer to operate on a less dense feature vector. So in our project, we replace the original residual block by bottleneck to increase the efficiency.

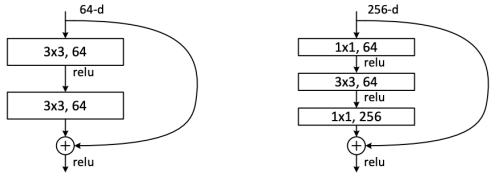


Figure 12: Left: a building block for ResNet. Right: a bottleneck building block for ResNet.

7.3 Bottleneck Upsampling

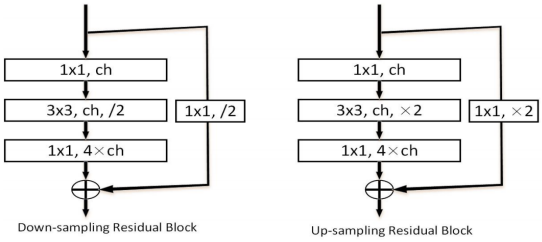
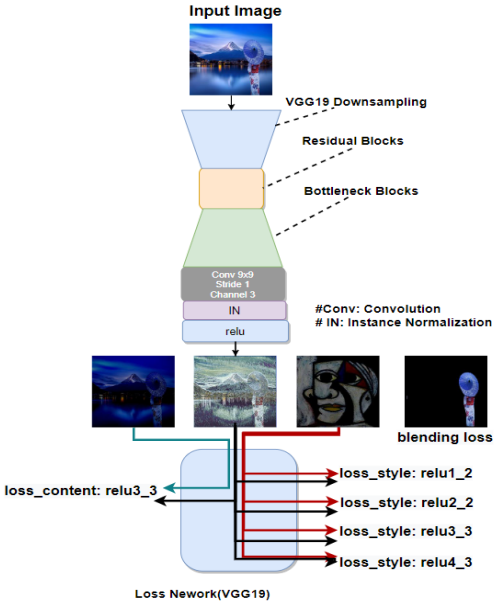


Figure 13: Zhang *et al* [15] extend the original down-sampling residual (left) to an up-sampling version (right).

7.4 Single-style Network Architecture

In our final single-style network, we use VGG19 for the downsampling part. Then, we use five residual blocks. Finally, we use bottleneck blocks for upsampling. Each convolutional layer is followed by instance normalization and ReLU nonlinearity.



Layer	Activation size
Input	256 x 256 x 3
VGG Downsampling	64 x 64 x 128
Residual block 1	64 x 64 x 128
Residual block 2	64 x 64 x 128
Residual block 3	64 x 64 x 128
Residual block 4	64 x 64 x 128
Residual block 5	64 x 64 x 128
Upsampling Bottleneck 1	128 x 128 x 128
Conv 1	128 x 128 x 64
Upsampling Bottleneck 2	256 x 256 x 64
Conv 2	256 x 256 x 3

Figure 14: Network architecture used for single-style style transfer networks.

For residual blocks, we adopt the same architectural design in the original network which is from Gross and Wilber [4] but we replace batch normalization with instance normalization. Each residual block has two 3x3 convolution layers with the same number of filters.

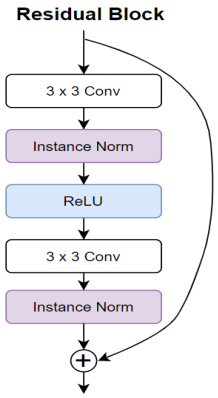


Figure 15: Residual block design in our networks.

7.5 Multi-style Architecture

7.5.1 MSG-Net

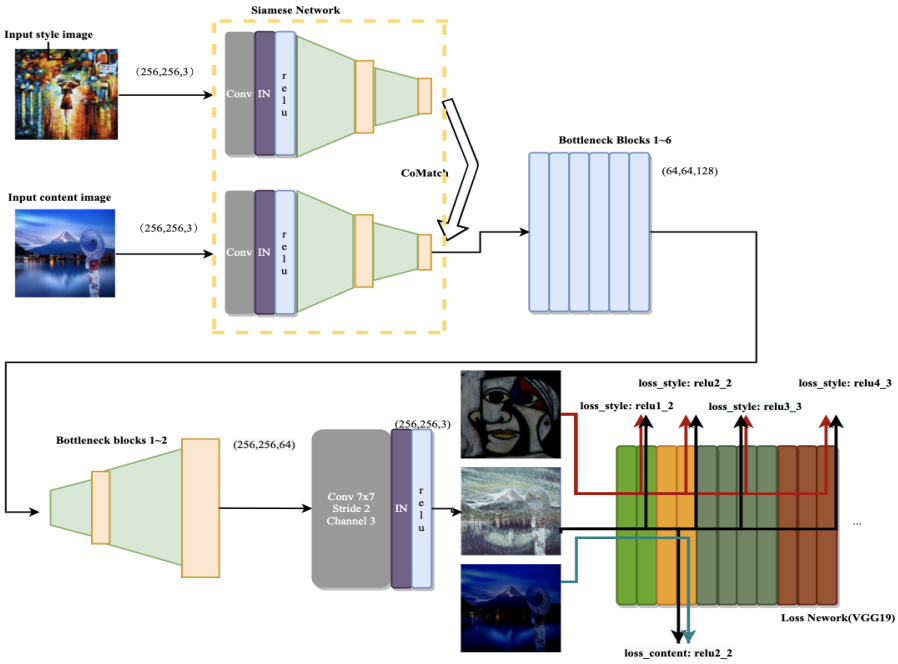


Figure 16: Original architecture of MSG-Net. This is the neural network architecture proposed by Zhang et al. We make modifications on the Siamese network, the six bottleneck blocks, and the loss function. See also Table 1 for details of size.

Table 1. Network architecture of MSG-Net.

Layer	Activation Size
Input: content	256 x 256 x 3
Input: style	256 x 256 x 3
Conv 1: content	256 x 256 x 64
Conv 1: style	256 x 256 x 64
Downsampling Bottleneck 1: content	128 x 128 x 128
Downsampling Bottleneck 1: style	128 x 128 x 128
Downsampling Bottleneck 2: content	64 x 64 x 512
Downsampling Bottleneck 2: style	64 x 64 x 512
CoMatch: content & input	64 x 64 x 512
Residual Block x 6	64 x 64 x 512
Upsampling Bottleneck 1	128 x 128 x 128
Upsampling Bottleneck 2	256 x 256 x 64
Conv 2	256 x 256 x 3

7.5.2 CoMatch Layer

As shown in Figure 17 for a given content image *Xc* and a style image *Xs*, the CoMatch layer attempts to preserve the semantic content of the feature map of Xc and matches the target style feature statistics(Gram Matrix):

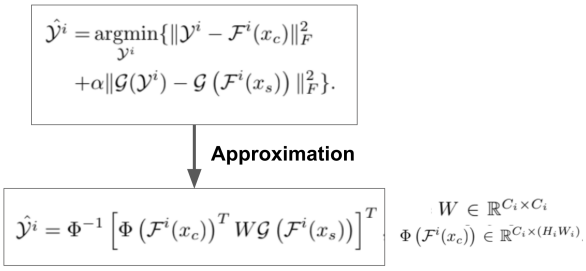


Figure 17: Formula of CoMatch initiated by Zhang *et al.*

where α is a trade-off parameter to balance the content and style weights. To achieve real-time style transformation performance, Zhang *et al* [15] introduce an approximation of the algorithm where W is a learnable weight matrix and Φ is a reshaping operation to match the dimension.

7.5.3 VGG Downsampling

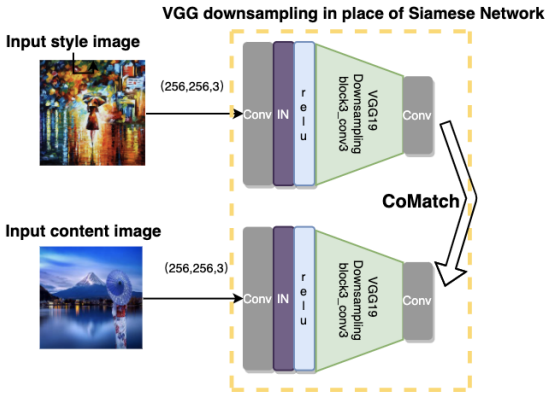


Figure 18: We used VGG19 Downsampling instead of the trainable Siamese network.

7.5.4 Multiple CoMatch while downsampling

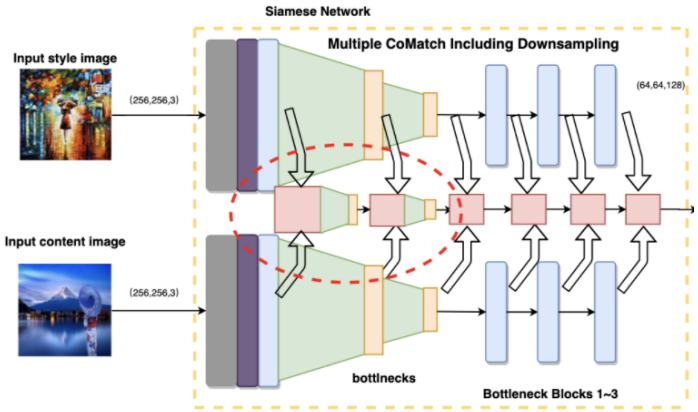


Figure 19: Multiple CoMatch applied in downsampling. The only difference between Multiple CoMatch and Multiple CoMatch including downsampling is the two CoMatch layers circled in red dots. Removing those layers can have the same effect as Multiple CoMatch. The squares in pink represent the CoMatch layer.

7.5.5 Unet

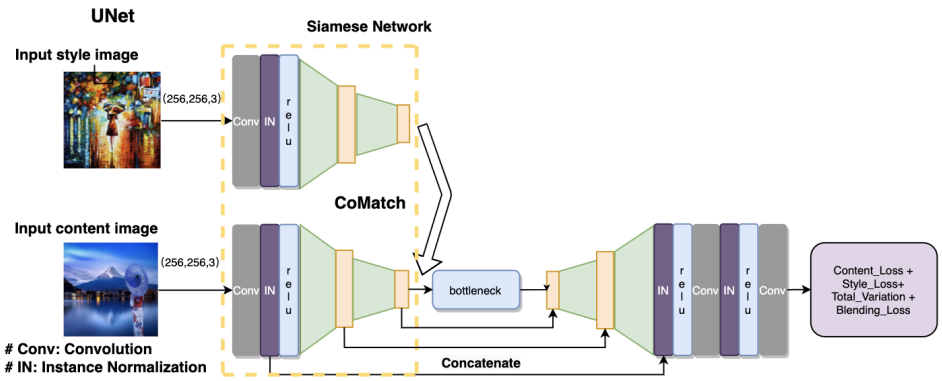


Figure 20: Multi-style network with blending loss using UNet architecture.

8 Contribution Summary

Individual Efforts

Sirui Hu:

- **Experimented Gatys' model in tensorflow with VGG19 and Resnet 50&152 in loss function and confirmed VGG19 as a better method than Resnet in style transfer.**
- **Researched Zhang's paper on the multi-style model, and migrated Zhang's MSG-Net code of multi-style model from MXNet to TensorFlow 2.**
- **Researched Mask RCNN and prepared the codes to generate complete masks for all the images in Microsoft COCO 2014 training Dataset. Coded and generated the percentage masks for all the images.**
- **Proposed blending loss, VGG downsampling, adding multiple comatch in the downsampling process of MSG-Net with confirmation from Weisheng and using UNet in the architecture with blending loss.**
- **Prepared the presentation with slides and scripts. Presented the slides and answered the questions in the QA session.**
- **Experimented with implementing VGG downsampling, Multiple CoMatch, UNet, using different methods to blend edges, blending loss with complete mask, and blending loss with percentage mask in the multi-style style transfer network.**

Weisheng Wang:

- **For single-style network architecture, found supporting reference paper[8] for VGG downsampling approach.**
- **For multi-style network architecture, proposed multiple comatch and found the reference paper[7] that uses similar network architecture.**
- **Researched most of the reference papers in order to find various approaches that could help us improve the performance of our networks.**
- **Experimented with implementing VGG downsampling and blending loss for the single-style style transfer network.**
- **Helped Sirui prepare for the presentation such as writing scripts and doing rehearsal; also answered some questions during the QA session.**

Kanglan Tang

- **Researched the paper of MSG-Net (Multi-Style transfer network) introduced by Zhang et al and rewrote the code of the network and the training function, which has been written in MXNet, using TensorFlow 2.0.**
- **Implemented the Multiple CoMatch layers on the Downsampling part of the Multi-Style transfer network and trained the model.**
- **Helped train some models of other experiments and the mask generating process, and prepared the presentation slides.**

Longxiang Dai

- **Wrote code in Python 3 using tensorflow 2.0. to implement the single-style network architecture introduced Johnson *et al***
- **Perused the reference paper about single-style and multiple-style networks, and explored how the architecture of multiple-style (such as bottleneck) could be used in single-style for better quality and efficiency**
- **Experimented with implementing VGG downsampling for the single-style transfer network**
- **Implemented the bottleneck and upsampling bottleneck architectures in the improved single-style transfer network**

Qian Zhao

- **Researched the paper of Multi-Style Transfer Network by Zhang et al. and A Neural Algorithm of Artistic Style by Gatys et al. as well as parts of blending and loss function.**
- **Experimented some parts of VGG and trained the model of Multi-style transfer network; prepared the presentation slides.**

Group Efforts

- **For the first 3 weeks, we switched our topic twice. At the beginning, we found and read several relevant papers for the recommendation system. However, after consulting Julian and Ruihan, we learned that traditional machine learning models could handle this task well. Then, we explored the music generation domain. However, none of us has a solid basis of knowledge in music theory. Finally, we turned to image style transfer because some of our team members have experiences with building deep learning models on images.**
- **Each member attended weekly group meetings to discuss progress and to distribute works.**

- **Sirui, Kanglan and Qian were responsible for the implementation of the multi-style network; Weisheng and Longxiang were responsible for the implementation of the single-style network.**
- **Sirui, Weisheng, Kanglan and Longxiang gathered the complete masks for all the images in the 2014 COCO training dataset.**
- **Sirui, Weisheng and Kanglan went to TA's office hour biweekly to consult Julian and Ruihan about our model and to ask for suggestions.**
- **All members participated in preparing the slides for the project presentation.**
- **All members participated in composing this final report paper while Weisheng and Sirui organized and revised the whole report before submission**