Controllable Real-Time Style Transfer

Yuting Sun
ytsun@stanford.edu

Xiangcao Liu
shawn610@stanford.edu

Abstract

We explored in this project how to train a real time controllable style transfer network. In particular, we implemented a style transfer neural network with two controllable input parameter: degree of style transfer and multiple style targets. The main idea is concatenating the parameters to the input image and adjusting the loss function accordingly so that the network is correctly optimized for different input parameters. The system we have implemented is able to transfer image to multiple styles in different degrees in a user controllable way.

1 Introduction

Image style transfer has been getting a lot of attention recently since Gatys et al. [1] has shown great success. In [1], Gram matrix-based losses is introduced to evaluate whether two styles are similar. An image is transferred into a new style through an optimizing process minimizing Gram-based loss between output image and ground-truth images. This expensive optimization is performed at test time and can generalize to any style target but is too slow. [2] proposed a real time model by training a feedforward convolutional neural network in a supervised manner, minimizing the loss between network output and ground-truth images. Compared to the optimization-based method, the pretrained network gives similar qualitative results but can be three orders of magnitude faster [2].

However, after such a network is trained, user can not specify the degree of style or different style targets. In this paper, We have implemented multiple techniques to train a controllable network. First, to train the network with controllable degree of style transfer λ, we concatenate the input image with random λ_{input}, and then apply the same λ_{input} to (4) to calculate total loss for back propagation. Secondly, to train the network for multiple styles, we use different style id as input and use this style image to calculate loss (4) in different batches. Thirdly, in order to efficiently train for the multiple styles, we used increment learning strategy and re-centered Gram matrix proposed by [3]. Using these techniques, our trained network can transform the input image to a target style specified by user with a degree also specified by user.

2 Related work

Fast image style transfer with controllable parameter has been studied previously. Mostly related to our work are [4], [3] and [5]. [4] attacked the problem of transferring to multiple styles from image reconstruction point of view without using any predefined style images. It allows a controllable degree of style transfer by applying the λ to content features during image reconstruction. The λ here dictates how much content to preserve after reconstruction. [5] implemented a controllable fast style transfer model which can control the stroke sizes.

[3] implement a multi-style transferring network in a controllable manner. It proposed multi-style transfer network by passing in a noise map activated for each style targets and used incremental learning strategy to generate a multi-style transferring network. It utilized several techniques to train effectively for multiple style targets.

Comparing to [3], we resorted to a simpler approach for multi-style task, simply passing in an image id and concatenating it to the input content images. We have implemented some of their training and optimization techniques in our project. We further built a more controllable network: User can control style target and degree of style transfer at the same time.

3 Dataset

Microsoft Common Objects in COntext (MS COCO) dataset [6] is a large scale data set containing 91 common object categories with 82 of them having more than 5,000 labeled instances. In total the dataset has 2,500,000 labeled instances in 328,000 images. We are using MS COCO 2017 dataset for our project, with train/val/test split to be 118k/5k/41k.

For style images, we use the BAM [7] dataset, which is a dataset of artistic images at the scale of ImageNet [8]. Each image in BAM is labeled with common object types, media types (i.e.,visual style) and emotion.

4 Technical methods

In this section, we first briefly introduce architecture of the non-controllable fast style transfer network from an open source implementation [9] following design in [2]. Then, we describe how we improve it into a controllable network.

4.1 Non-controllable Fast Style Transfer Model

The fast style transfer network consists of a trainable image transformation network and a pre-trained loss network that is used to define several loss functions.

4.1.1 Transformation network Architecture

The input are color images of $256 \times 256 \times 3$ which will be passed to a transformation network consisting of 3 convolutional layers, 5 residual blocks, and 2 fully connected layers. The output of this transformation network is also a $256 \times 256 \times 3$ color image $\hat{y}$.

4.1.2 Loss Network and Loss Function

A 19-layers VGG network [10] pretrained on ImageNet dataset [8] is used as a loss network to compute content loss and style loss. We denote the loss network as ϕ. $\phi_l(C)$ is the activations (also called feature map) of layer l of the network with input C. Here, C is the input content image, $\hat{y}$ is the image generated by transformation network, and S is the input style image

The content loss is calculated in (1):

$$L_{content} = \|\phi_l(C) - \phi_l(\hat{y})\|^2 \tag{1}$$

The style loss is calculated in (2):

$$L_{style} = \left\|G_l^{\phi}(S) - G_l^{\phi}(\hat{y})\right\|_F^2 \tag{2}$$

As described in [2], a total variation regularizer L_{tv} is used to encourage spatial smoothness in the output image. Total loss is a weighted combination of loss functions:

$$L = \gamma \|\phi_l(C) - \phi_l(\hat{y})\|^2 + \lambda \left\|G_l^{\phi}(S) - G_l^{\phi}(\hat{y})\right\|_F^2 + L_{tv} \tag{3}$$

4.2 Baseline Controllable Model

4.2.1 Updated Input

In order to train a controllable network, we updated the inputs and loss functions. As show in figure 1, our style transformation network are trained with 4 inputs: $256 \times 256 \times 3$ content images C, a fixed $256 \times 256 \times 3$ style image S, a random integer λ_{input} between (0, 100), a random integer $styleId$ between (0, $NumOfStyles$). At both training and test time, λ_{input} and $styleId$ will be expand to a $256 \times 256 \times 1$ matrix and concatenate to each content image to become C_{concat}. C_{concat} will go through the same network mentioned in previous section.

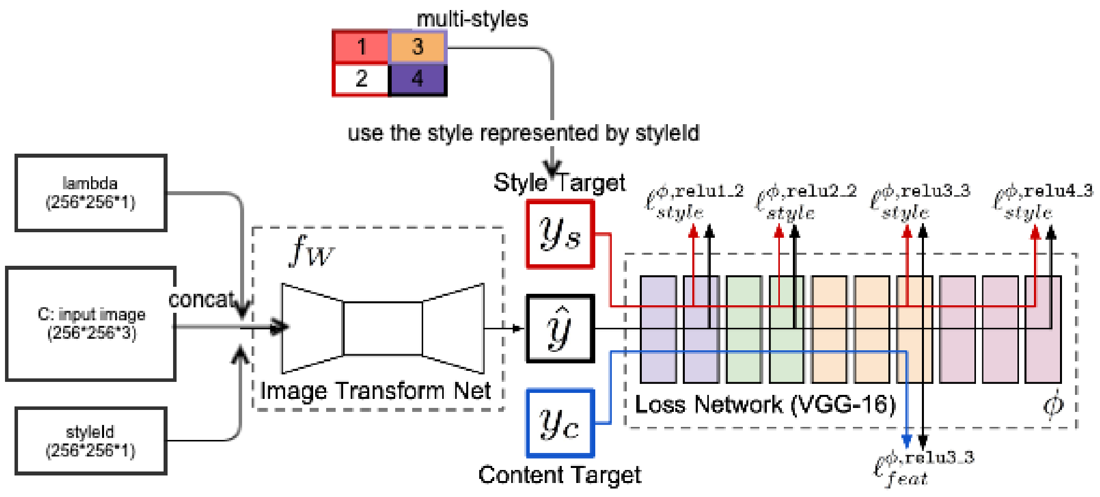
multi-styles
1
3
2
4
use the style represented by styleId
lambda (256*256*1)
C: input image (256*256*3)
styleId (256*256*1)
concat
f_W
Image Transform Net
Style Target
y_s
$\hat{y}$
y_c
Content Target
$\ell_{style}^{\phi,relu1\_2}$
$\ell_{style}^{\phi,relu2\_2}$
$\ell_{style}^{\phi,relu3\_3}$
$\ell_{style}^{\phi,relu4\_3}$
Loss Network (VGG-16)
ϕ
$\ell_{feat}^{\phi,relu3\_3}$

Figure 1: Network with controllable input parameters

4.2.2 Updated Loss Function

Since the goal of our model is to train a network with a controllable degree of style during test time, we applied the input parameter λ_{input} on L_{style} to enforce the network has a smaller style loss when λ_{input} is large.

Hence our updated total loss will be:

$$L = \gamma \left\| \phi_l(C) - \phi_l(\hat{y}) \right\|^2 + \lambda_{input} \left\| G_l^{\phi}(S_{styleId}) - G_l^{\phi}(\hat{y}) \right\|_F^2 + L_{tv} \tag{4}$$

where $S_{styleId}$ represent one of the style images.

4.3 Improved Controllable Network

The initial results we achieved with our baseline model with controllable input parameters is not satisfactory and needs further improvement. This section describes the optimizations we have implemented.

4.3.1 Incremental Training

We have tried sampling different style images in each mini-batch but the results are very poor (see Figure 3). This is likely because all optimizations learned in previous mini-batches will be overwhelmed in next mini-batch after switching to a new style images. We therefore adopted the incremental training strategy similar to what is described in [3]. The idea is we will only add new style into the training after we have learned well with existing styles. This is process is manual currently, as we visually look at the transformed images and decide whether to add one more style.

4.3.2 Mean Subtracted Gram

Since we are training with multiple styles, the scale of gram matrix of different target styles could vary significantly. Therefore, we adopted the idea from [3], to re-center the activations by subtracting mean before their inner product. This will prevent losses and gradients from different styles changing drastically.

$$\overline{G_l} = (\phi_l(C) - \overline{\phi_l(C)})(\phi_l(C) - \overline{\phi_l(C)})^T \tag{5}$$

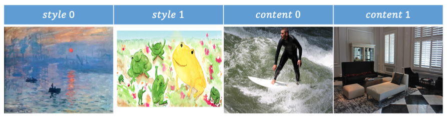

Figure 2: Style and content input

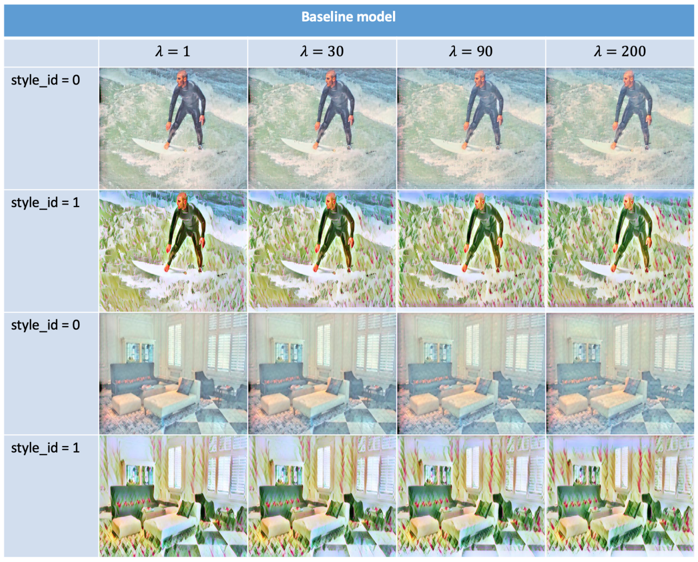

Figure 3: Visual results of generated images with baseline controllable network

5 Results

Our Tensorflow [11] based implementation can be found here at github: [12]. After our network is trained, we tested with inputs shown in Figure 2.

The result for our baseline controllable model is shown in Figure 3. As we can see, increasing λ is not generating images with stronger style as expected. Style blended as images on third row display some color pattern from from style 1 while we should only see features from style 0.

Figure 4 shows the results after we retrained using incremental Training. The transformation is more sensitive to λ. However, there are still issues of style blending.

Figure 5 presents the results with updated Gram by subtracting from its mean and shows better relevance to both λ and $styleId$. This means style blending earlier is likely due to different scale of gram matrix of styles.

6 Future

New loss function The model we have trained so far is based on independent γ weight on content loss and λ weight on style loss. One idea we would like to try is to assign γ = 100 - λ.

Concatenating λ to later layers Instead of concatenating controllable λ to the input image, we want to experiment concatenate the lambda to later layers to be more sensitive to λ changes.

Different network architecture We also would like to try different network architecture, like more number of layers, residual blocks, etc as we feed more number of styles into the network to adjust for the complexity needed for mapping to different styles.

7 Contributions

Yuting Sun: Project definition; Literature study; Data collection; Data processing, Experiment design; Baseline implementation; Architecture study; Error analysis and algorithm tuning; Milestone writeup; Poster writeup; Poster session recording; Final writeup.

Xiangcao Liu: Project definition; Literature study; Data processing,Experiment design; Baseline implementation; Architecture study; Error analysis and algorithm tuning; Milestone writeup; Poster writeup; Poster session recording; Final writeup.

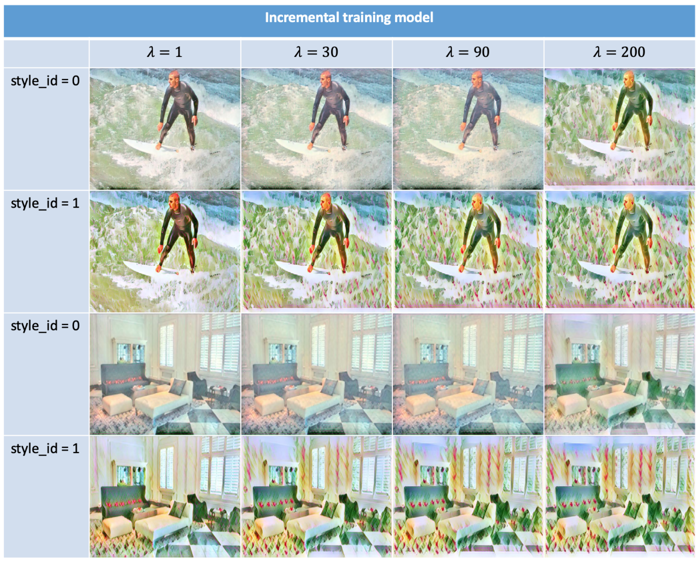

Figure 4: Visual results of generated images with controllable network using incremental training strategy

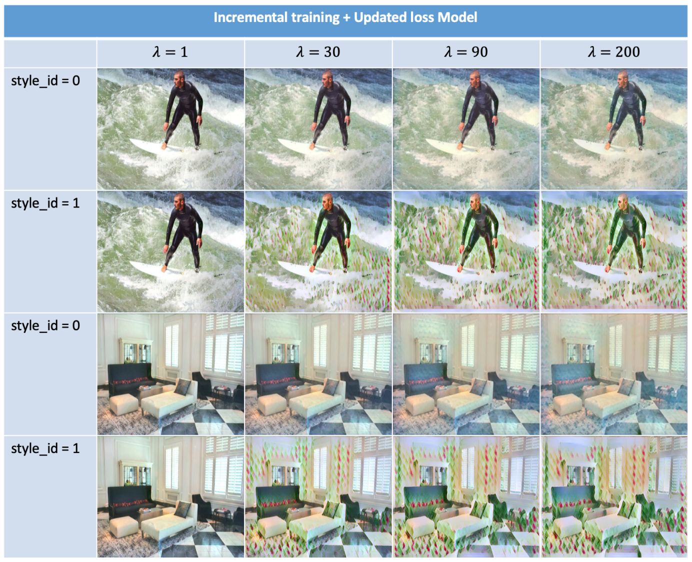

Figure 5: Visual results of generated images with controllable network using incremental training strategy and mean-substracted Gram matrix

References

[1] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A Neural Algorithm of Artistic Style. *arXiv e-prints*, page arXiv:1508.06576, Aug 2015.

[2] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, 2016.

[3] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Diversified Texture Synthesis with Feed-forward Networks. *arXiv e-prints*, page arXiv:1703.01664, Mar 2017.

[4] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal Style Transfer via Feature Transforms. *arXiv e-prints*, page arXiv:1705.08086, May 2017.

[5] Yongcheng Jing, Yang Liu, Yezhou Yang, Zunlei Feng, Yizhou Yu, Dacheng Tao, and Mingli Song. Stroke Controllable Fast Style Transfer with Adaptive Receptive Fields. *arXiv e-prints*, page arXiv:1802.07101, Feb 2018.

[6] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common Objects in Context. *arXiv e-prints*, page arXiv:1405.0312, May 2014.

[7] Michael J Wilber, Chen Fang, Hailin Jin, Aaron Hertzmann, John Collomosse, and Serge Belongie. Bam! the behance artistic media dataset for recognition beyond photography. *arXiv preprint arXiv:1704.08614*, 2017.

[8] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[9] Logan Engstrom. Fast style transfer. `https://github.com/lengstrom/fast-style-transfer/`, 2016. commit xxxxxxx.

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

[11] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[12] Ytsun. Controllable fast style transfer. `https://github.com/tingsun/fast-style-transfer/`, 2019. commit xxxxxxx.