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 \( \lambda \), we concatenate the input image with random \( \lambda_{\text{input}} \), and then apply the same \( \lambda_{\text{input}} \) 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], [5] and [3]. [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 \( \lambda \) to content features during image reconstruction. The \( \lambda \) 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 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 dataset, which is a dataset of artistic images at the scale of ImageNet. 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 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 pretrained on ImageNet dataset is used as a loss network to compute content loss and style loss. We denote the loss network as $\phi$. $\phi_l(C)$ 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):

$$L_{\text{content}} = \|\phi_l(C) - \phi_l(\hat{y})\|^2_2$$

(1)

The style loss is calculated in (2):

$$L_{\text{style}} = \left\| G^\phi_l(S) - G^\phi_l(\hat{y}) \right\|^2_F$$

(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:

$$L = \gamma \|\phi_l(C) - \phi_l(\hat{y})\|^2_2 + \lambda \left\| G^\phi_l(S) - G^\phi_l(\hat{y}) \right\|^2_F + L_{tv}$$

(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 shown in figure 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 $\lambdainput$ between (0, 100), a random integer $styleId$ between (0, NumOfStyles). At both training and test time, $\lambdainput$ 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.
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 $\lambda_{\text{input}}$ on $L_{\text{style}}$ to enforce the network has a smaller style loss when $\lambda_{\text{input}}$ is large.

Hence our updated total loss will be:

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

where $S_{\text{styleId}}$ 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.

$$\bar{G}_l = (\phi_l(C) - \bar{\phi}_l(C))(\phi_l(C) - \bar{\phi}_l(C))^T \quad (5)$$
Figure 2: Style and content input

![Baseline model]

<table>
<thead>
<tr>
<th>Baseline model</th>
<th>$\lambda = 1$</th>
<th>$\lambda = 30$</th>
<th>$\lambda = 90$</th>
<th>$\lambda = 200$</th>
</tr>
</thead>
<tbody>
<tr>
<td>style_id = 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>style_id = 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>style_id = 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>style_id = 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 $\lambda$ 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 $\lambda$. 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 $\lambda$ 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 $\gamma$ weight on content loss and $\lambda$ weight on style loss. One idea we would like to try is to assign $\gamma = 100 - \lambda$.

**Concatenating $\lambda$ to later layers** Instead of concatenating controllable $\lambda$ to the input image, we want to experiment concatenate the lambda to later layers to be more sensitive to $\lambda$ 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-subtracted Gram matrix
References


