CSE 599 G1 Project Report
Mangafy: Neural Style Transfer

Tun Sheng Tan
netid: tunsheng
tunsheng@uw.edu

1 Introduction

Manga and anime are popular forms of reading around the world with their captivating stories and art styles. It takes talent and experience to create the aesthetic which is inaccessible to majority of the population. Developing a tool to convert images into manga scene could potentially enable a way of creating manga content and increase productivity of illustrators.

Gatys et al. [1] pioneered the use of neural network for style transfer, which is now the gold standard. By using a convolution neural network (CNN), they are able to generate images with the desired content but with a different artistic style. Their approach has been the backbone for more sophisticated methods like photorealistic neural style [2], CycleGAN [3] and CartoonGAN [4]. However, Gatys et al neural style does not always preserve the features such as the edge which is an important feature in manga and anime. On the other hand, CartoonGAN has been successful at cartoonizing photos with clear edges but such an approach requires retraining the network with a new database.

In this project, our goal will be to introduce modifications to Gatys neural style to promote edges and to create a new database with anime/manga style images for training a CartoonGAN. We proposed some modifications to Gatys neural style to promote the edges in the styled images. We also implemented a foreground-background segmented style transfer based on Mask-RCNN[5]. In addtion, with the success of network such as CartoonGAN at tranferring cartoon style, we have created our own dataset consisting of manga and anime styled images to attempt at generating manga styled images.

2 Related Work

Gatys neural style [1] is the gold standard for style transfer. This method introduces two novel representation for style and content. Instead of comparing pixel differences, it compares high level features from a VGG network [6]. To capture the style, it uses a Gram matrix which captures the correlation between the features from different layers of the VGG network.

The use of a Gram matrix introduces training instability and representability issues. Gram-based methods require parameter fine-tuning to have stable optimization. In addition, it is found that the Gram matrix is not unique for different feature maps, which leads to the instability [7]. CartoonGAN overcomes these problems by using a neural network for style representation. CartoonGAN uses a generator to create cartoon-like images to fool a discriminator [4]. The discriminator is trained to classify real photos from cartoons. This network architecture has similar properties to CycleGAN [3] where training can be done with unpaired images. CartoonGAN has been successful in producing high resolution cartoonize image with clear edges.

In addition to CartoonGAN, several authors proposed semantic segmentation using markov random field (MRF) based style transfer to preserve content [8, 9]. The segmentation maps generated by MRF are used to refine style transfer for different semantic objects. Apart from MRF, Faster-RCNN [10] is another popular segmentation model. Faster R-CNN passes the feature maps from a CNN to a Region Proposal Network to generate a set of bounding boxes which will be resized by pooling. Finally, a fully connected layer will return the corresponding bounding boxes for each object.

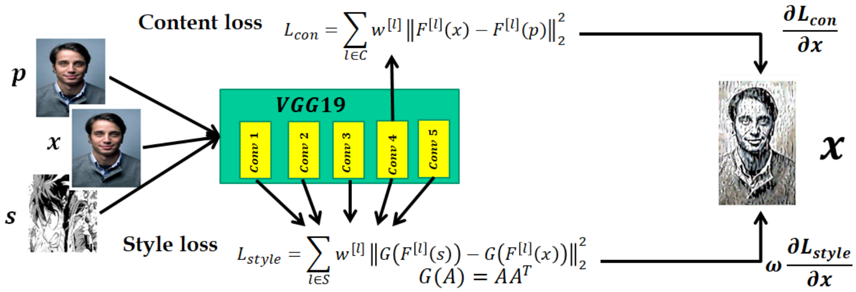

Figure 1: Gatys neural style. The style image (s), content image (p) and output image (x) are passed through VGG network. Content loss (L_{con}) and style loss (L_{style}) are computed using features from different layers of VGG network. x is then updated according to the gradients of $L_{con} + \omega L_{style}$.

3 Methods

3.1 Neural Style Transfer

The style transfer problem based on Gatys neural style approach [1] is to convert a noisy input x into an image with the content of image p and with the style of image s. An overview of neural style is shown in Figure 1. More formally, suppose that the l^{th}-layer feature map of an image x from a pre-trained network is denoted by $F^l(x) \in \mathbb{R}^{C_l \times N_l}$ where C_l is the number of channels at l^{th}-layer and N_l is the size of the feature map and W is the width of the feature map. The style representation based on the Gram matrix G is denoted by $G(F^l(x)) \in \mathbb{R}^{C_l \times C_l}$:

$$G(F^l(x)) = \Big(F^l(x)\Big)\Big(F^l(x)\Big)^T \tag{1}$$

The style transfered image x is the solution to the following problem:

$$\begin{aligned} x &= \arg\max_{x'} \; L_{total}(x', p, s) && (2) \\ &= \arg\max_{x'} \; \alpha L_{con}(x', p) + \beta L_{style}(x', s) && (3) \\ &= \arg\max_{x'} \; \alpha \sum_{l\in\mathcal{C}} w_l \Big\| F^l(x') - F^l(p) \Big\|_2^2 + \beta \sum_{l\in\mathcal{S}} w_l \Big\| G\big(F^l(x')\big) - G\big(F^l(p)\big) \Big\|_2^2 && (4) \end{aligned}$$

where α and β are hyperparameters, w_l are the weights, $\mathcal{C}$ is the set of feature map layers of trained network used for representing the content and $\mathcal{S}$ is the set of feature map layers of trained network used for representing the style. Following Gatbys et al., we let the weights w_l be the same for all the terms and choose the fourth layer of VGG-19 as the content map and all the layers of VGG-19 as the style map. To solve the minimization problem, we use BFGS optimizer.

3.2 Feature promoting modification

A content loss function using a L_2 norm difference between the original image and the generated image tends to generate artifacts on the generated image. This is because L_2 norm promotes small but non-zeros contributions for all the features even if they are irrelevant. Therefore, we proposed the use of a L_1 norm which is known to promote sparsity among features.

$$L_{con}(x', p) = \sum_{l\in\mathcal{C}} w_l \Big\| F^l(x') - F^l(p) \Big\|_2^2 \tag{5}$$

In addition, the Gram matrix can be susceptible to noises in the correlation. Thus, L_2 norm difference will also result in a rough texture. To this end, we also enforce L_1 norm difference to generate a smoother texture.

$$L_{style}(x', s) = \sum_{l\in\mathcal{S}} w_l \Big\| G\big(F^l(x')\big) - G\big(F^l(p)\big) \Big\|_2^2 \tag{6}$$

While transfering style to the entire picture is desirable, sometimes a segmented style transfer is more desirable as in the case of portraits. Segmented style transfer can also provide a more refined control of style transfer while preserving details in the content. Therefore, we proposed the use of Mask-RCNN to segment the images and apply neural style transfer to different segments. Mask-RCNN is an extension of Faster R-CNN which generates the same output and also the segmentation masks [5]. In our method, we apply the segmentation masks to separate the subject from the background before performing style transfer as shown in Figure 2.

Figure 2: Style transfer with Mask R-CNN (MRCNN). The content image is passed through MRCNN. The masks from MRCNN are applied to the content to produce a subject and a background image. Then, neural style is applied to the subject with a targeted style. The output image is generated by combining the styled subject with the background image.

3.3 CartoonGAN

CartoonGAN has been successful at cartoonizing a photo [4]. The advantages of CartoonGAN over Gatys neural style are clearer edges and smoother color shadings. Due to the similarity of cartoon and anime/manga style, CartoonGAN is an ideal candidate model to achieve anime/manga style transfer for photos. CartoonGAN consists of two CNNs, a generator (G) and a discriminator (D). In this architecture, G is trained to generate a cartoonized photo to fool D while D is trained to classify real photos and cartoons. D is trained to discriminate cartoons (c) from real photos (p) and edge-smoothed cartoons (e). The purpose of introducing edge-smoothed cartoons is to promote edges in the generated image. An overview of CartoonGAN design is shown in Figure 3.

The loss function for CartoonGAN consist of two parts: a edge-promoting adversarial loss L_{adv} and a content preserving loss L_{con}.

$$L(G,D) = L_{adv}(G,D) + \omega L_{con}(G,D) \tag{7}$$
$$L_{adv}(G,D) = \log[D(c)] + \log[1-D(e)] + \log[1-D(G(p))] \tag{8}$$
$$L_{con}(G,D) = \sum_{l\in\mathcal{C}} w_l \|F^l(G(p)) - F^l(p)\|_1 \tag{9}$$

where ω is a hyper-parameter that controls amount of content infromation in the generated image. The adversarial loss L_{adv} which plays the role of style loss in Gatys neural style is applied to G and D while the content loss L_{con} is only applied to G. The edge-smoothed cartoons (e) are generated by Gaussian smoothing the dilated edges generated from a Canny edge dector. Similar to Gatys neural style, CartoonGAN uses the same VGG network layer for feature extraction.

4 Experiments

4.1 Neural Style

We wrote neural style from scratch based on Gatys algorithm. Since neural style is an online based method, there is no need for a training dataset. The baseline for our comparison is Gatys neural style (with L_2 norm for both content loss and style loss). We performed optimization for our baseline and our models using the same parameters. For our

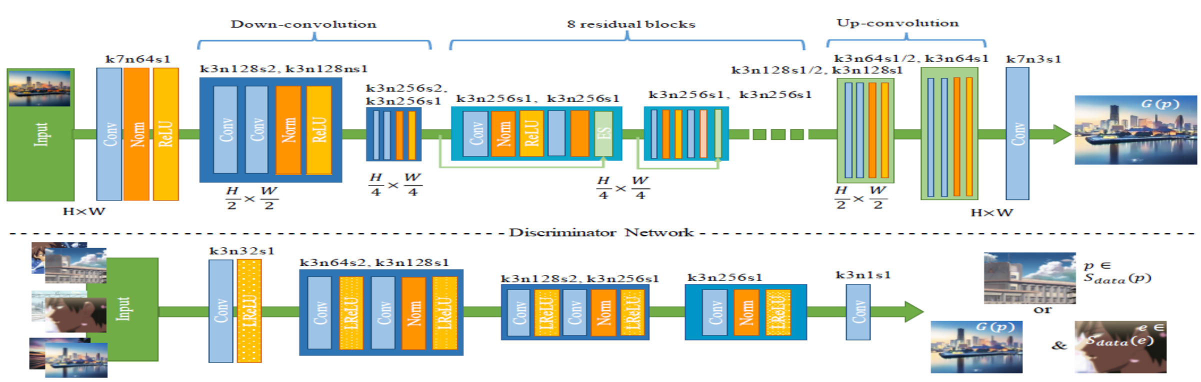

Figure 3: CartoonGAN architecture for the generator (top) and the discriminator (bottom) networks [4]. The parameters for the convolution layer (Conv) are labeled by kernel size k, output channel n and stride s. Batch normalization layer is labeled *Norm* and elementwise sum is labled *ES*.

experiment, we used 300 steps with content to style weight $\alpha : \beta$ ratio of 10^{-6}. The result for our models are shown in Figure 4. The baseline produces visually pleasing images but the edges for certain objects are not clear and there are artifacts in the shading. Model 1 (L_1 content loss) shows improvement in the edges as shown in the cat image but no significant improvement in the dog images. Model 2 (L_1 style loss) generates an image with a clear edge and a lower noise. This is apparent in the mouth region of the dog and cat. Model 3 is a combination of Model 1 and 2. The resulting image has smoother shading as shown in the background of the cat image and the dog image. Model 4 is Model 3 applied to Mask R-CNN output. The resulting image is a styled subject with real background.

Running neural style is straight forward with little problem. The average time for an interation takes about 1 to 10 seconds depending on the size of the input image. For each run in our experiment, it takes about 1 minute per image for 300 iteration steps. We notice that for certain style, Gatys neural style produces wiggling artifacts as shown in Figure 5.

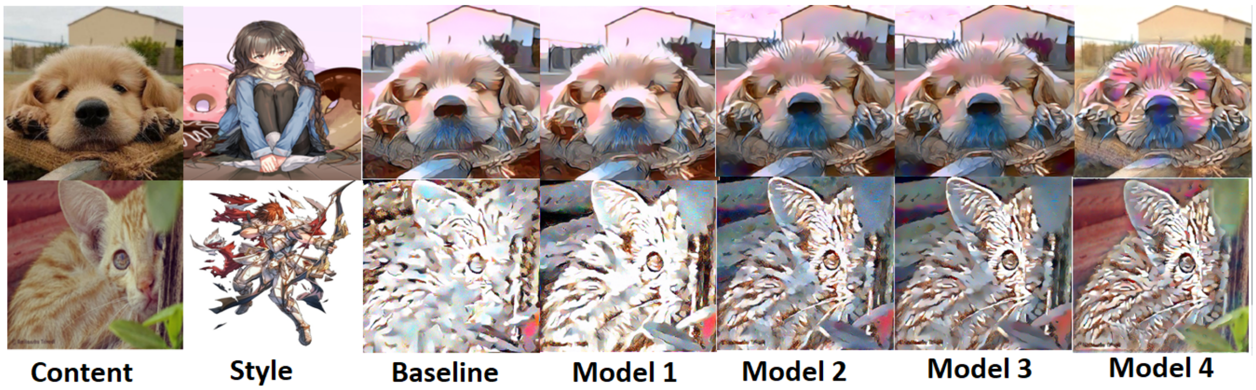

Figure 4: Comparison between L_1 and L_2 normed loss. Given a content image and a style image, neural style transfer are performed for 5 different models. (**Baseline**) L_{style}: L_2, $L_{content}$: L_2 (**Model 1**) L_{style}: L_2, $L_{content}$: L_1 (**Model 2**) L_{style}: L_1, $L_{content}$: L_2 (**Model 3**) L_{style}: L_1, $L_{content}$: L_1 (**Model 4**) L_{style}: L_1, $L_{content}$: L_1, with **Mask R-CNN**.

Figure 5: (Left) Content image (Middle) Style image (Right) Styled content image using Gatys neural style with artificial wiggling lines.

4.2 Generating Dataset and CartoonGAN

In order to generate manga and anime style images from CartoonGAN, we need a dataset with the relevant style. We scraped 4200 manga style images from anime-pictures.net using imgbrd-grabber. Since we are interested in portrait images, Instagram was a good source of such images. We scraped 4200 animal and human portrait images from Instagram using instagram-scraper. In order to reduce the size of images and to ease training, we converted the images to a size of 256x256 in Hierarchical Data Format.

While creating the dataset, we had difficulty getting the desired manga styled images due to copyright issues and the time constraint. We underestimated the time required to remove dialog bubbles from manga. A neural network based solution could speed up this process which is a potential future project. Therefore, we relaxed the requirement for manga style images to include anime styled images.

We wrote our own CartoonGAN from scratch. During the data loading stage on Google Colab, we experienced difficulty in loading the dataset. Our batch size is limited to 10 images due to memory constraint. Downsizing the images from 256x256 is not a possible solution due to degrading in style content. We pretrained out CartoonGAN to reproduce the original image with 10 epochs. Then, we proceed with actual training. Training failed because the GAN generates the input image with a color filter applied to it as show in Figure 6. Training on Colab takes 1 hour per epoch which limits our ability to do a lot of experiments.

Figure 6: (Left) Content image (Right) Styled content image using CartoonGAN. CartoonGAN failed to generate manga style image.

5 Conclusion

In this project, we proposed and showed that sparsity in high level features promotes edges and lowers artifacts in style transfer. We also implemented a foreground-background segmented style transfer based on Mask-RCNN. In addition, we created a dataset of manga/anime styled images and a dataset of portraits for training a CartoonGAN. However, we failed to train the CartoonGAN.

We plan to implement a multi-scale segmentation style transfer with support for real time style transfer. Finally, we plan to retrain CartoonGAN using our datasets with consultation from GAN experts.

References

[1] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. [1, 2]

[2] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. Deep photo style transfer. *arXiv preprint arXiv:1703.07511*, 2017. [1]

[3] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017. [1]

[4] Yang Chen, Yu-Kun Lai, and Yong-Jin Liu. Cartoongan: Generative adversarial networks for photo cartoonization. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. URL `http://openaccess.thecvf.com/content_cvpr_2018/html/Chen_CartoonGAN_Generative_Adversarial_CVPR_2018_paper.html`. [1, 3, 4]

[5] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2017. [1, 3]

[6] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014. [1]

[7] Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, Yizhou Yu, and Mingli Song. Neural style transfer: A review, 2017. [1]

[8] Huihuang Zhao, Paul L. Rosin, and Yu-Kun Lai. Automatic semantic style transfer using deep convolutional neural networks and soft masks, 2017. [1]

[9] Alex J. Champandard. Semantic style transfer and turning two-bit doodles into fine artworks, 2016. [1]

[10] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2015. [1]