[COMP4471] Manga Style Transfer

Lu Weiqi, Yi Sophia, Zheng Hantao

Figure 1. Jojo's Bizarre Adventure, a Japanese cartoon series popular all over the world for decades

Abstract

Generative adversarial networks(GANs) have demonstrated their effectiveness in image synthesis. And recent works have shown GAN's potential in style transfer, more specifically, cartoon stylization. The application of the cosmic style transfer could not only provide the social media users with a new way to share their daily lives. Additionally, this technology generates manga pictures instantly, which can be a handy tool to inspire manga artists to create more interesting mangas. In this project, we explored different GAN models and applied CartoonGAN to a manga of very unique art style named JoJo's Bizarre Adventure. After face cropping the cartoon images and training them with 2 different setups of hyper-parameters. We successfully generated videos and images of JoJo's style, our model is able to produce high-quality images and videos visually.

1. Introduction

With the revolutionary advent of style transfer, stylized images of different art forms, ranging from line scratch to classical painting can be synthesized directly from content images and style images. Given the popularity and broad application of animation, cartoon stylization has been a heated topic in the area. However, different manga series can have very distinctive art styles. Many current implementations only generalize photos to a general cartoon style, i.e., thin edge, smooth colour shading, and high level of abstraction. When it comes to specific cartoon styles, the framework is not able to produce very satisfactory results.

In this project, we investigate and explore different models and seek to apply a framework to provide satisfactory performance in transforming real-world photos into the art style of JoJo's Bizarre Adventure. Known for its thick angular lines, the sharper appearance of character, dark shadow and detailed drawing, JoJo's Bizarre Adventure attracts hundreds of millions of readers worldwide and has a unique style of drawing. We experimented with different sets of hyper-parameters on CartoonGAN [1] to generate images and videos in JoJo's manga style. Although CartoonGan is a previous work proposed for a similar task, it is not tailored specifically for the manga stylization of a specific series. It is interesting to see the quality of generated videos when the model is fed by cartoon scenes of JoJo and real-world photos. And our results have demonstrated that the model could transform the colourization of JoJo's image effectively and naturally.

2. Related works

We compared different style transfer methods to obtain the most suitable framework to be applied to the proposed problem:

2.1. Neural Style Transfer and Fast Style Transfer

Style transfer is the task of recomposing the style of an image into the domain of another. Greyls et al [2] introduced the idea and proposed a Neural Algorithm of Artistic Style to separate and recombine the content and style of an image. Taking a content image A and a style image B as input, an image is generated to contain the semantic content of A and the style of B. The image representations are derived from deep Convolutional Neural Networks optimised for object recognition, which make high-level image information explicit.

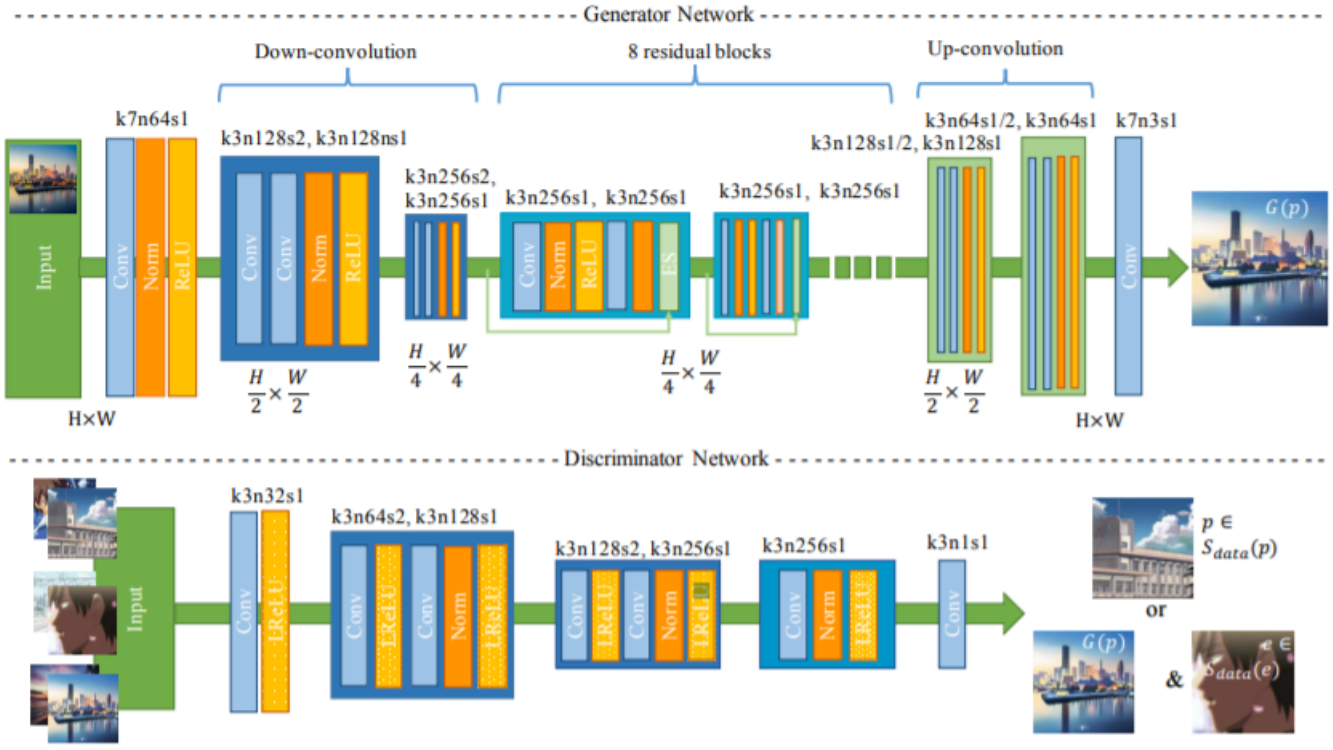

Figure 2. [1] Architecture of the generator and discriminator networks in the proposed CartoonGAN, in which k is the kernel size, n is the number of feature maps and s is the stride in each convolutional layer, 'norm' indicates a normalization layer and 'ES' indicates elementwise sum.

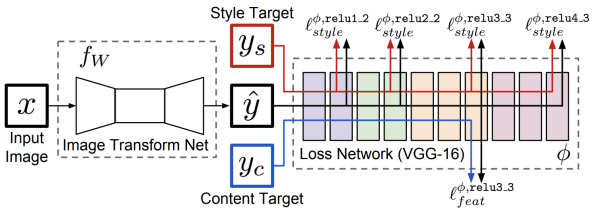

Figure 3. [5] Fast Style Transfer network architecture

However, the neural style transfer approach is computationally expensive because each image requires many forward and backwards passes through deep CNN networks such as VGG. Johnson et al [5] introduced an alternative approach to improve the efficiency of training. The idea is to train a feed-forward network for each style, and use pre-trained CNN to compute the loss. After training, a single feed-forward network would perform the style transfer. The process is illustrated in figure 1.

Though the Fast Style Transfer method improves the efficiency of training, the performance is not as satisfactory in manga style transfer, as the cartoon images are simplified and abstract compared to real-world images such that their texture features are relatively hard to extract from a single input style image. Hence this approach is not appropriate to address our proposed problem.

2.2. GANs and CycleGAN

Generative Adversarial Networks (GANs) is another promising approach to generate stylized images. The model consists of two networks, a generator network to generate images, and a discriminator network to distinguish between real and synthesized images [4]. The generator network is trained to fool the discriminator such that the adversarial loss provided by the discriminator pushes the generated images close to the target manifold.

It is worth noting that when using GANs or fast style transfer to generate new images, paired image sets are always required. Each content image needs to have its corresponding style image, which is hard to obtain. To address the problem, CycleGAN [10] was proposed to perform style transfer with unpaired data. It trains two sets

of GAN models simultaneously, mapping the images from class A to class B and from class B to class A, respectively.

But this approach is also limited by the fact that CycleGAN requires the concurrent training of two GAN models. The training process of two networks is challenging and expensive, making the result unstable.

3. Data

The training data includes both cartoon scenes and real-world photos and the test data only consists of real-world photos. For the cartoon image, we captured 4000 images from the stop motion animation of the first season of JoJo's Bizarre Adventure. And 18234 photos of human faces were downloaded from CelebA [6]. For preprocessing, we converted each image to a tensor with the RGB channels in the range [0.0, 1.0] then resized and cropped them to 256×256. The cartoon images are face cropped to facilitate the transition in styling. Then they are edge-smoothed [1] by:

1. detecting edge pixels with a Canny edge detector.
2. dilating the edge regions stylization.
3. applying a Gaussian smoothing in the dilated edge regions

For training, we shuffle all the images and randomly divide the training dataset into training set and validation set with a proportion around 9:1. In this process, we gathered 3600 training samples, 400 validation samples. And for testing we had 50 test samples randomly taken from CelebA.

4. Method

So eventually, we chose CartoonGAN to be our starting point of the project. CartoonGAN utilizes a GAN model to learn the mapping between photo and cartoon manifolds using unpaired training data, and is able to generate cartoon images efficiently [1].

Similar to CycleGAN, CartoonGAN formulates the process of learning to transform real-world photos into cartoon images as a mapping function that maps the photo manifold P to the cartoon manifold C. A discriminator network D is trained to distinguish images in the cartoon manifold from other images and provide the adversarial loss for Generated image G so as to push the generated images G similar to the cartoon manifold. Let L be the loss function, G* and D* be the weights of the networks.

The objective is to solve the min-max problem:

$$(G*, D*) = \arg\min_{G} \max_{D} L(G, D) \tag{1}$$

Since the GAN model is highly nonlinear, the optimization can be easily trapped at local minimum or saddle point with random initialization. To alleviate such problems and accelerate the convergence of the style transfer network, we will reconstruct real-world images in pre-trained phase.

4.1. Loss Function

The loss function $L(G, D)$ in Equation (1) above consists of two parts:

1. the adversarial loss that drives the generator network to generate images similar to the cartoon manifold.
2. the content loss, which preserves the image content during cartoon stylization.

4.1.1 Adversarial Loss

The adversarial loss is applied to both generator and discriminator. Its value indicates to what extent the synthesized image is similar to the manga style.
For each image $c_i \in S_{data}(c)$, the following steps [1] are applied:

1. detect edge pixels with a Canny edge detector.
2. dilate the edge regions stylization.
3. apply a Gaussian smoothing in the dilated edge regions.

The edge smoothing ensures that the discriminator would not be easily confused by the generated images that are with correct shading but wrong edges.

The edge-promoting adversarial loss is defined as:

$$\begin{aligned} L_{adv}(G, D) = &\mathbb{E}_{c_i\ S_{data}(c)}[logD(c_i)] \\ &+ \mathbb{E}_{e_j\ S_{data}(e)}[log(1 - D(e_j))] \\ &+ \mathbb{E}_{p_k\ S_{data}(p)}[log(1 - D(G()p_k))] \end{aligned} \tag{2}$$

4.2. Content Loss

In the meantime, we also need to ensure the resulting manga images retain the semantic content of the input photos. The feature maps in the VGG network [8] pre-trained by [7] have been demonstrated to have good object preservation ability. Accordingly, we define the content loss as:

$$L_{con}(G, D) = \mathbb{E}_{p_i\ S_{data}(c)}[||VGG_l(G(p_i)) - VGG_l(p_i)||] \tag{3}$$

A simple addition is used for the loss function:

$$L_{adv}(G, D) = L_{adv}(G, D) + \omega L_{con}(G, D) \tag{4}$$

ω is the weight to balance the two given losses. Larger ω leads to the preservation of more content information from the input photos, pushing our stylized images to obtain a more detailed texture. However, if the generated image has a very detailed texture, it would be less cartoonized since the key characteristic of the art style of cartoon is abstraction. Alternatively, low ω may also be problematic since

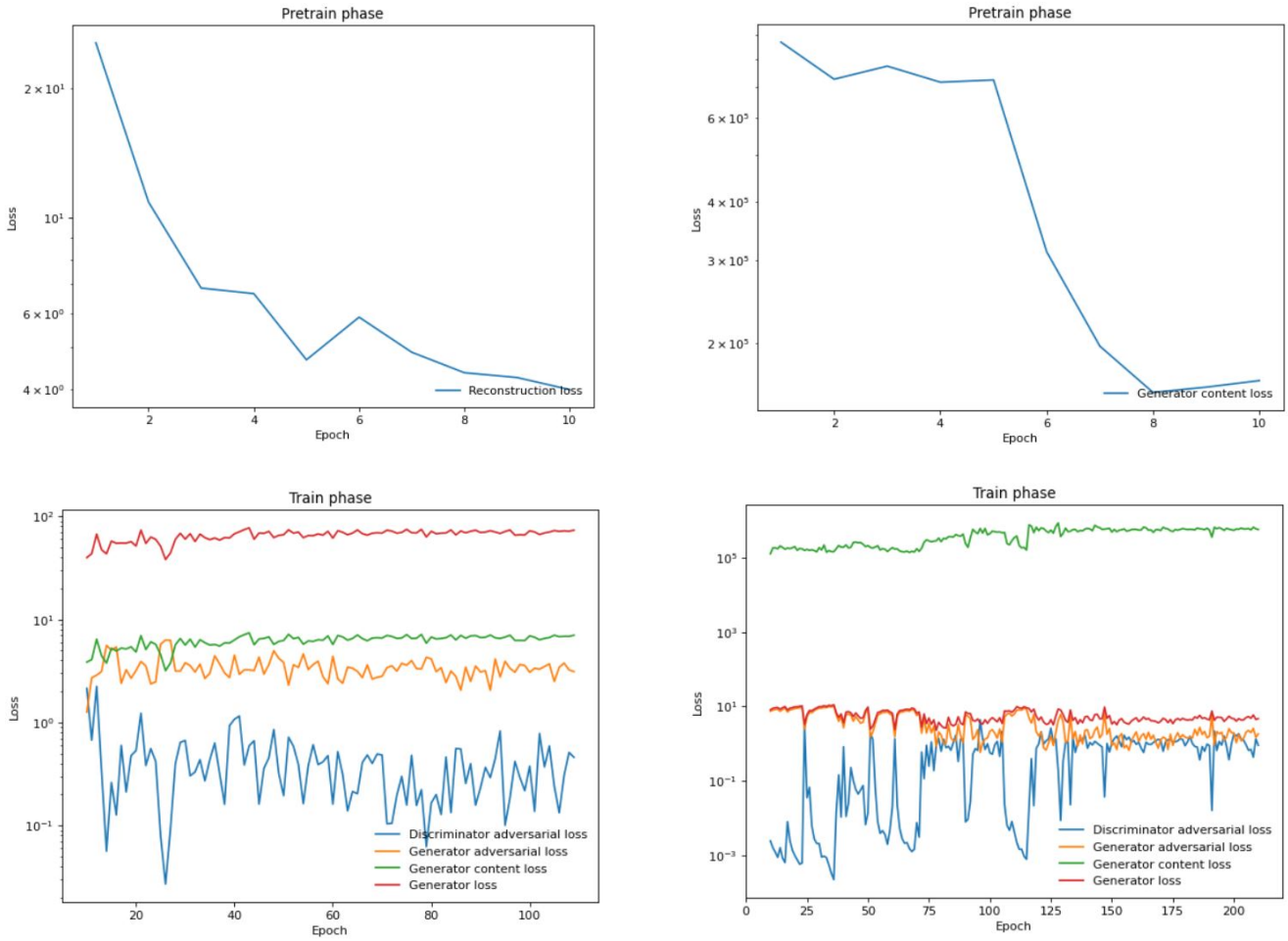

Figure 4. **Loss** Pre-train loss and training loss graph for both settings

it would lead to the loss of semantic content of the image. Hence, we need to experiment with different ω to make informed decisions when dealing with the trade-off between the content loss and style loss.

5. Experiments

We experiment with the CartoonGAN model using two different setups of hyper-parameters and compare the outcomes to obtain a framework of satisfactory performance on cartoon stylization (Source code: link). The trained models in our experiments are saved to enable the evaluation of models visually. All experiments were performed on GPU provided by GoogleColab.

To prevent over-fitting, during training, we save the checkpoints for 210 epochs and input the 50 test images to each of the saved models. We visually compare each set of the synthesized test images and determine that the checkpoint at 209 outperforms all the models. Thus we choose the checkpoint saved at epoch 209 as our final model.

5.1. Hyper-parameters Comparison

The architectures of two implementations are uniform as both are based on CartoonGAN. And the hyper-parameters of the implementations are shown in Table 1, the setups are almost identical. The number of epochs the model was trained on are the same: 210 epochs. Both setup are with small batch sizes, i.e.16 and 10 to obtain faster convergence. The learning rate for the generator and the discriminator are the same, which is 2e-4. The optimizer applied is the Adam optimizer, the beta1 for the discriminator and the generator of GAN are set to 0.5 instead of the default value 0.9. Obtaining a smaller beta1 is helpful in training GANs to prevent the discriminator from learning too fast [3]. If the discriminator loss drops to zero, the generator gradient would vanish, leading to the failure of

Input image | Hyperparameter setting 1 | Hyperparameter setting 2

Figure 5. Results of the trained CartoonGAN under two different hyperparameter settings.

learning for the generator.

The main difference between the setups is that setting 1 is with instance normalization and VGG19 and setting 2 is with batch normalization and VGG16. And there is a significant discrepancy between the content loss constant loss , which are 10 and 0.00005 respectively. To monitor the learning process of our models, model losses were plotted against epochs and batches. As shown in Figure 4, both models demonstrate a steady trend of convergence in the pre-train phase, and the curves for training are similar.

Figure 5 shows the comparison between the outcome of the models in human faces. From the result we could see that the implementation with batch normalization outperforms the instance normalization visually by having closer art style to JoJo's and more natural colorization of human faces. Probably due to the fact that the first setup utilized a less deeper VGG net for feature mapping, which is a more optimal feature extractor for the generator.

Figure 6. **Failure** Failures in some input images under hyperparameter setting 1, There are strange objects such as eyes occur in the generated images

5.2. Common Failures

In the initial training stage, we trained the model with Flickr30k [9] image set and cartoon images that have not been face cropped. Despite the smooth transition in the background, we noticed that there are "eyes", as shown in Figure 6, and white point appearing randomly in the synthesized pictures. The problem is due to the fact the model required a consistent match between inputs from the real-photo training set and cartoon image training set. To alleviate this mismatch between the real-photo image and cartoon images, we re-trained the model with CelebA datasets and face-cropped cartoon characters image. The performance indeed improves. The main figures are smoothly transformed to the art style of JoJo's without any discrepancy like "eyes". But the background is slightly distorted and is not as satisfactory as the output of the model trained with Flickr30k images as shown in the videos in the supplementary section. Thus, to remedy the potential failure in training, a consistency in the inputs is required to train the model.

Hyperparameter	Description	Setting 1	Setting 2
batch size	size of a single batch	10	16
beta1	param of adam optimizer	0.5	0.5
beta2	param of adam optimizer	0.999	0.999
con lambda	weight of content loss	10	0.000005
input size	size of the input images	256	256
lrD	learning rate of the discriminator	0.0002	0.0002
lrG	learning rate of the generator	0.0002	0.0002
# pretrain epochs	number of pretrain epochs	10	10
# train epochs	number of training epochs	100	200
vgg model	vgg models used	vgg19	vgg16

Table 1. **Settings** – two sets of hyperparameters settings

6. Conclusions

In this paper we applied CartoonGAN to transform real-world photos into the art style of JoJo's Bizarre Adventure.

Aiming at generating features of JoJo's, we explored different inputs and two setups of parameters to fine-tune the model. The experiments show that CartoonGAN can learn a model that transforms photos of real-world scenes to cartoon style images with high quality and high efficiency i.e. 400s per epoch of training while we choose a human face dataset and face-cropped cartoon image scenes and apply batch normalization and VGG16 instead of instance normalization and VGG19.

7. Future works

Our current work only performs the transformation of color between the images. In future work, we would like to investigate how to to improve cartoon stylization so that the characters in the synthesized image would have unique contour lines and shading in their faces like JoJo's. It would also be interesting if the CartoonGAN is applied to other manga series with distinctive art styles like Pokemon, Doraemon to perform the stylization in real-life photos.

Other possible future extensions and new applications could be the addition of some simple image processing techniques after the forward pass. It might significantly improve the image quality output by the cartoonGAN. By applying to sharpen and adjusting the contrast and saturation appropriately, it is possible that the color mapping is made to be more reasonable to a large extent. Therefore we can add a few image processing steps after the forward pass of our trained generator to further strengthen the manga style of the image.

It is also considerable to slightly modify the loss function to shift our training objective's attention a bit. We can also extract the edge patterns as additional features to imitate manga style and hence improve the edge detail of our generated images.

Another promising approach might be to further improve the quality of our training dataset. The current training set is roughly face-cropped and resized using a face detection tool. The performance is not very satisfactory on the anime image dataset. As a consequence, some faces in the original images are missing while some processed images contain non-image objects, which leads to instability and difficulty in the training phase. Therefore, it can be foreseen that using anime specified face detectors to do a face-cropping job will be very helpful to improve the quality of our dataset.

References

[1] Chen et al. CartoonGAN: Generative Adversarial Networks for Photo Cartoonization. In *CVPR*, 2018. 1, 2, 3

[2] Gatys et al. Image Style Transfer Using Convolutional Neural Networks. In *CVPR*, 2016. 1

[3] Goodfellow et al. Improved techniques for training gans. In *NeurIPS*, 2016. 4

[4] Ian Goodfellow et al. Generative Adversarial Nets. In *NeurIPS*, 2014. 2

[5] Johnson et al. Perceptual Losses for Real-Time Style Transfer and SuperResolution. In *ECCV*, 2016. 2

[6] Liu et al. Deep Learning Face Attributes in the Wild. In *ICCV*, 2015. 3

[7] Russakovsky et al. Imagenet large scale visual recognition challenge. In *IJCV*, 2015. 3

[8] Simonyan et al. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*, 2015. 3

[9] Young et al. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. 2014. 6

[10] Zhu et al. Unpaired imageto-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017. 2

[COMP4471] Manga Style Transfer

Supplementary Material

A. Enviornment description

All the model training were performed on Google colab pro virtual machine environment with cuda graphic cards equipped. The following are the python dependencies used:
Python 3.7.12
jupyter 1.0.0
matplotlib 3.2.2
pandas 1.1.5
tensorboard 2.7.0
pytorch 1.10.0+cu111
torchvision 0.11.1+cu111
numpy 1.19.5

B. Result Demo

URL link for transfer result, application on video and source code:
link