Multi-Style Cartoonization on Real-World Videos using CartoonGAN

李孟霏 F74076205
楊逸婷 F74077112
許盛雯 F74062036
李彥儒 F74066420

Abstract

In this paper, we are going to introduce our project with the implementation of Fast Style Transfer and CartoonGAN. Our datasets are collected from anime movies in different styles, and segmentate the frames into independent photos, which are easier to use for later training. And we use the COCO dataset for testing.

Keywords— cartoonization, styling, real-world

1. Introduction

Cartoon is a drawing, a simple example of creative thinking, and perception of reality and dream. In a few lines, the cartoonist can capture the entire idea he wants to convey, to observe and exaggerate the characters'features. Many famous cartoon images are created based on real-world scenes. However, drawing high-quality cartoon images will require a tremendous amount of time. Recently, leaning-based style transfer methods on artificial intelligence have drawn considerable attention. Although there are a variety of methods have been developed to create images with flat shading, mimicking cartoon styles. They use either image filtering or formulations in optimization problems. Such methods are too complicated to capture rich artistic styles. Thus in this paper, we propose Fast Style Transfer and GartoonGAN for photo cartoonization. From the perspective of computer vision algorithms, the goal of cartoon stylization is to map images in the photo manifold into the cartoon manifold while keeping the content unchanged. To achieve this goal, we propose to use a dedicated architecture and this will be introduced in Part 3 System Works later.

2. Motivation

After seeing the video that turned a real person's face into cartoon style, we got interested in this technique. The url is https://www.youtube.com/watch?v=5ZKzN8HtOkE&ab_channel=%E5%85%AD%E6%8C%87%E6%B7%B5Huber. We want to implement multi style cartoonization on real-world videos and bring value to people who need it.

2.1. Creating a virtual background for MetaVerse

As we all know, the idea of MetaVerse is getting popular. Our model can help build a virtual world's background where people can choose which style of universe they want to stay in according to their preference.

2.2. To help painter reducing their workload

Traditional animated movies run at 24 frames (think 1 drawing = 1 frame) per second of screen time. That's 1400 frames per minute and 86400 frames per hour of screen time. This resulted in a huge investment in time and effort. Thus, creators can rely on this model by transforming the real world picture into their own style, relieving their burden on works.

2.3. To help people creating their own video

Nowadays, photo shooting has become a common behavior. We can notice there are many applications and

photo editing software that have created an amusing filter or AR which can overlay the digital content and information into the physical world. With the help of this model, people may produce their own video and obtain great fascination.

3. System Framework

To achieve our goal of transforming real video into cartoon style, we have implemented 2 methods including Fast Style Transfer and CartoonGAN. We will explain what we have done and make an introduction on models'framework.

3.1. Fast Style Transfer

First, we choose some scene in the Pixar movie as our transformation style target.The training dataset for content is 4000 photos of people in 2017 COCO dataset. On the

Figure 1. images in training set

basis of the paper of Johnson et al [2], Model's framework is divided into 2 parts Figure 2 . The first part is image transform network which consists of Deep Residual Network Figure 3. Input is image to convert X.Output is converted image Y which combines style and content. The second

Figure 2. system overview

part is loss network which consists of pretrained VGG-19 model.We replace VGG-16 from the paper.The model has

Layer	Activation size
Input	3 × 256 × 256
Reflection Padding (40 × 40)	3 × 336 × 336
32 × 9 × 9 conv, stride 1	32 × 336 × 336
64 × 3 × 3 conv, stride 2	64 × 168 × 168
128 × 3 × 3 conv, stride 2	128 × 84 × 84
Residual block, 128 filters	128 × 80 × 80
Residual block, 128 filters	128 × 76 × 76
Residual block, 128 filters	128 × 72 × 72
Residual block, 128 filters	128 × 68 × 68
Residual block, 128 filters	128 × 64 × 64
64 × 3 × 3 conv, stride 1/2	64 × 128 × 128
32 × 3 × 3 conv, stride 1/2	32 × 256 × 256
3 × 9 × 9 conv, stride 1	3 × 256 × 256

Figure 3. image transform network

well-trained image filter.We get high level features to calculate loss Figure 4. We calculate content loss on only one layer Figure 5 while calculating style loss on all the layers Figure 6

Content Loss: $L^l_{content} = ||\frac{1}{C_l H_l W_l}(F_l(\hat{y}) - F_l(y))||_2^2$

Style Loss: $L^l_{style} = ||\frac{1}{C_l H_l W_l}(G_l(\hat{y}) - G_l(y))||_2^2$

MSELoss: $L_{pixel} = \frac{1}{CHW}||\hat{y} - y||_2^2$

total variation loss: $L_{tv} = Mean||\sum_{h,w}(\hat{y}_{h+1,w} - y_{h,w}) + (\hat{y}_{h,w+1} - y_{h,w})||_2^2$

=> Loss: $L = \lambda_{style}L_{style} + \lambda_{content}L_{content} + \lambda_{pixel}L_{pixel} + \lambda_{tv}L_{tv}$

Figure 4. loss function

Figure 5. content output from the loss network layers

Using a pretrained model saves a lot of time. It takes less than 4 seconds to run the Fast Style Transfer model.

3.2. CartoonGAN

CartoonGAN is based on the structure of GAN. GAN consists of two models, namely the generator and the dis-

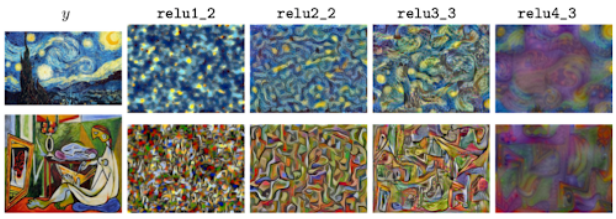

Figure 6. style output from the loss network layers

criminator. In the case of CartoonGAN [1] , the discriminator classifies input images as cartoon or non-cartoon images. On the other hand, the generator takes real-world images as inputs and aims to generate cartoonized outputs that are able to fool the discriminator.

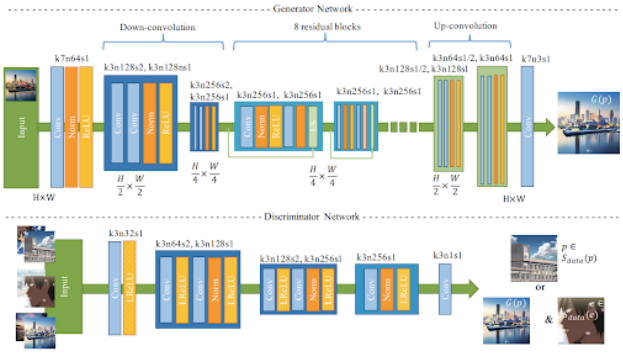

Figure 7. CartoonGAN network architecture

3.2.1 Loss Function

Our objective is to solve the min-max problem $argmin_G max_D L(G, D)$. The loss function consists of two parts: (1) the adversarial loss $L_{adv}(G, D)$ and (2) the content loss $L_{con}(G, D)$. We calculate the total loss by addition of the two terms. $L_{adv}(G, D)$ represents how similar

$$\mathcal{L}(G, D) = \mathcal{L}_{adv}(G, D) + \omega \mathcal{L}_{con}(G, D),$$

the discriminator inputs look to the desired cartoon style. One important feature of cartoon images is that they have clear edges. However, the edges only take a little portion of the whole image. The clear edge feature is easily overlooked. A generated image with correct shading but unclear edges is likely to fool the discriminator. To avoid this problem, a dataset called "cartoon-smoothed"is created. This dataset contains cartoon images that go through edge smoothing filters. This dataset is labeled as non-cartoon for discriminator classification. Thus, $L_{adv}(G, D)$ is defined as $S_{data}(c)$ is the cartoon dataset, which contains cartoon

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

images from artists of the desired style. $S_{data}(e)$ is the cartoon-smoothed dataset. $S_{data}(p)$ is the photo dataset, which contains real-world photos that will be converted to cartoonized images. In addition to synthesizing real-world photos with cartoon style, the semantic content of real-world photos needs to be preserved as well. $L_{con}(G, D)$ represents the semantic content difference between real-world photos and the outputs of Generator. We obtain the content loss using the high-level feature map on a pretrained VGG model. l refers to the "conv4_4"layer of the VGG

$$\begin{aligned}&\mathcal{L}_{con}(G, D) = \\ &\quad \mathbb{E}_{p_i \sim S_{data}(p)}[||VGG_l(G(p_i)) - VGG_l(p_i)||_1]\end{aligned}$$

network.

3.2.2 Dataset

We collected 7,382 stills of Studio Ghibli movies as cartoon dataset, and 4,000 photos of people in the COCO dataset as real-world photo dataset.

4. Results

4.1. Fast Style Transfer

We pick three movie scenes as our Pixar's style target. Fig 8, 9, and 10 show our result.The style of color is similar to our target and people in the photo are smooth like Pixar's characters.

4.2. CartoonGAN

We pick three real-world photos from COCO dataset and transform them into anime photos in the style of Studio Ghibli using cartoonGAN model. Fig 11, 12, and 13 show our result. It can be found that some obvious animation

Figure 8. result from Luca

Figure 9. result from Turning Red

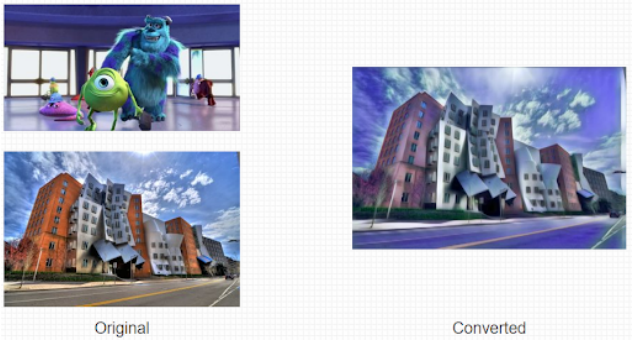

Figure 10. result from Monster Inc.

features, such as black borders around important objects, and soft colors are produced.

4.2.1 FLOPs and Parameters

FLOPs : 61.73G
Parameters: 11.13M

Figure 11. result 1 of CartoonGAN

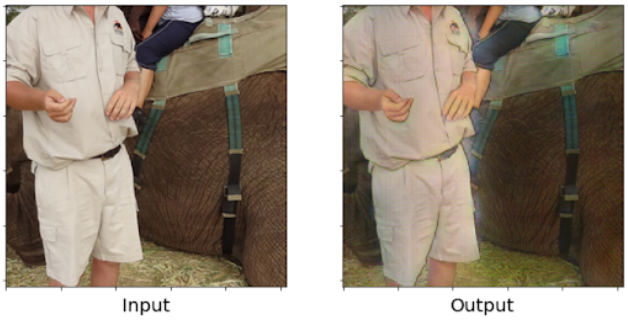

Figure 12. result 2 of CartoonGAN

Figure 13. result 3 of CartoonGAN

4.3. Time Performance

As we had tried to train on two different models, we came out the efficiency of the two models to convert images was different.

Fast Style Transfer	cartoonGAN
1.578125 sec	0.27062 sec

Figure 14. Time consuming to transform one photo (Test image size:256x256)

5. Demonstration

We implemented our training model by using the QT interface on the Windows system.

Figure 15. UI interface

video link

fox.mp4

original: https://drive.google.com/file/d/12kZDxAn8cAgbr9VKDSCop-MagL9vjvEe/view?usp=sharing
transfered: https://drive.google.com/file/d/1wzid9dVkHpzclSzgUTmV1YC0d-sRRhrH/view?usp=sharing

skateboard.mp4

https://drive.google.com/file/d/1RAFLHmOniz3Bsu5L-tDKWsA_soFAlcZw/view?usp=sharing

6. Conclusions and Future Works

Aiming at transforming real-world photos into cartoon style images have reached a successful accomplishment. The experiments showed that Fast Style Transfer and CartoonGAN are able to finish the job but with different FLOPs and time consuming. In the future work, we decide to implement some facial extraction on this model and make it more functional when it comes to cartoon stylization on human faces. Although we had added a gaussian noise layer before the discriminator to make the d_loss change from zero, we believe that there will be more improvements we can make. We will try similar ideas for the entire network layer, which will investigate further.

7. Requirements

You will need the following to run the above: - TensorFlow
- Pytorch
- Colab
- If your computer didn't have GPU, it will consume plenty of time for training

References

[1] Yang Chen, Yu-Kun Lai, and Yong-Jin Liu. Cartoongan: Generative adversarial networks for photo cartoonization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9465–9474, 2018. 3

[2] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution, 2016. 2