IMAGE TO CARTOON CONVERSION

Mini Project Report

Submitted by

Gopika V Baburaj

Reg No : FIT20MCA-2054

Submitted in partial fulfillment of the requirements for the award of the degree of

Master of Computer Applications
Of
A P J Abdul Kalam Technological University

FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)®
ANGAMALY-683577, ERNAKULAM(DIST)
FEBRUARY 2022

DECLARATION

I, **Gopika V Baburaj** hereby declare that the report of this project work, submitted to the Department of Computer Applications, Federal Institute of Science and Technology (**FISAT**), Angamaly in partial fulfillment of the award of the degree of Master of Computer Application is an authentic record of our original work.

The report has not been submitted for the award of any degree of this university or any other university.

Date :

Place: Angamaly

FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)®

ANGAMALY, ERNAKULAM-683577

DEPARTMENT OF COMPUTER APPLICATIONS

CERTIFICATE

This is to certify that the project report titled **"Image to Cartoon Conversion"** submitted by **Gopika V Baburaj** towards partial fulfillment of the requirements for the award of the degree of Master of Computer Applications is a record of bonafide work carried out by them during the year 2022.

Project Guide **Head of the Department**

Submitted for the viva-voice held on ..... at .....

ACKNOWLEDGEMENT

I am deeply grateful to **Dr. Manoj George** , Principal, FISAT, Angamaly for providing us with adequate facilities, way and means by which we were able to complete our mini project work and I express my sincere thanks to **Dr. C Sheela** ,Vice Principal FISAT, Angamaly.

Our sincere thanks to **Dr. Deepa Mary Mathews**, Head of the department of MCA, FISAT, who had been a source of inspiration. I express heartiest thanks to **Ms. Senu Abi** our project guide for their encouragement and valuable suggestion . We express our heartiest gratitude to our scrum master **Ms. Manju Joy** and the faculty members in our department for their constant encouragement and never ending support throughout the project.We would also like to express our sincere gratitude to the lab faculty members for their guidance

Finally we express our thanks to all my friends who gave me wealth of suggestion for successful completion of this project.

ABSTRACT

Cartoons are artistic form widely used in our daily life. Manually recreating real-world scenes in cartoon styles is very laborious and involves substantial artistic skills.So creating a software that can convert real world images easily to artistic cartoons will be of great use.

In this image to cartoon conversion using GAN (Generative Adversarial Network) and CNN (Convolutional Neural Network), we can generate high-quality cartoonized images from real-world photos. Images are decomposed into three cartoon representations: the surface representation, the structure representation, and the texture representation. Corresponding image processing modules are used to extract three representations for network training, and output styles could be controlled by adjusting the weight of each representation in the loss function. Here the model is trained with a cartoon image to learn its features and then convert every real world image given as input to cartoon

Contents

Chapter 1

Introduction

Cartoon is a popular art form that has been widely applied in diverse scenes. Some famous products have been created by turning real-world photography into usable cartoon scene materials, where the process is called Image Cartoonization. It has application in various fields such as entertainment, graphics etc. So, developing a system that can automatically convert an image to cartoon will be of great use.

Converting real world images to painting like cartoons is of large popularity now a days. There exist many software and applications that are used for this purpose now. But most of them have some limitations as the rate of conversion, features of image may decrease.This project aims to study how to improve the rate of clarity in conversion without losing the original features of the image.

Chapter 2

PROOF OF CONCEPT

Stylizing images in an artistic manner has been widely studied in the domain of non-photo realistic rendering. Traditional approaches develop dedicated algorithms for specific styles. However, substantial efforts are required to produce fine-grained styles that mimic individual artists. In this cartoonization project, we develop a model that is capable to convert real world images to cartoon without losing majority of its properties.

2.1 Objectives

To develop a model that will convert any given random input image to cartoonized image

- To identify the features of the original image .
- To converting the original image features to cartoon image features
- To save or download the cartoon image.

Chapter 3

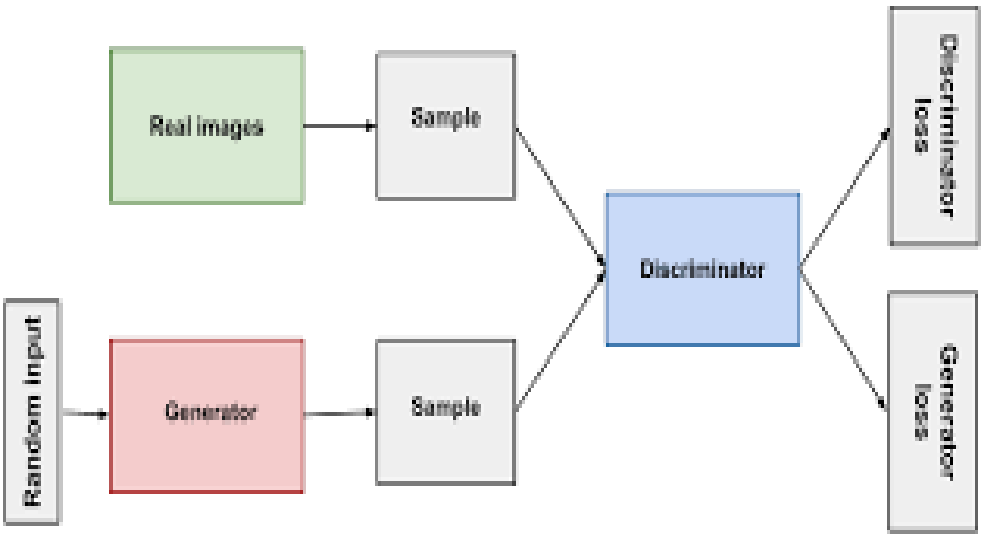
IMPLEMENTATION

.

In this project, three common features of cartoon images are observed from investigation of cartoon images and artistic paintings. 1. Global structure is composed of sparse color blocks 2. The surface are smooth with little textures 3. Details are represented by lines and boundaries Based on these observations, this system extract three white box cartoon representations and optimize them end to end in a GAN frame work. .

Image processing modules extract each representation and a Generative Adversarial Network (GAN) framework is used to learn the extracted representations and cartoonize the input images. Output styles can be controlled by adjusting the weight of each representation in the loss function. It uses GAN framework, but with one generator and two discriminators where the first discriminator aims to distinguish between the surface representation of the outputs and the original cartoons. The second one is used for the same goal replacing the surface by the texture of the images. Among the three features, surface representation is very important. Its role is to imitate the cartoon painting style. The second principal component they used is the structure. Generator extracts the high level features to enforce special constrain between the results and the extracted structure rep-

resentation. Finally. the texture representation is used in the form of a random color shift algorithm to extract single channel texture representation from color images. That means it retains high frequency texture information and decreases the influence of color and luminance to make the results look more cartoonish.



3.1 Dataset

Human face and landscape data are collected for generalization on diverse scenes. For real-world photos, we collect 10000 images from the FFHQ dataset for the human face and 5000 images from the dataset in for landscape. For cartoon images, we collect 10000 images from animations for the human face and 10000 images for landscape.During training, all images are resized to 256*256 resolution, and face images are feed only once in every five iterations.

3.2 Modules

3.2.1 FEATURE STUDY OF TRAINING IMAGES

We will have a set of original images and cartoon images for training. We make the system study about the features of both the real world images and cartoon images.

3.2.2 IMAGE PREPROCESSING

F The main step of image preprocessing is resizing of images and extracting required features of the image. The size of the output image will always be 256 X 256. The features extracted at this stage are

Surface representation

Structure representation

Texture representation

3.2.3 CARTOONIZATION OF THE FEATURES

SURFACE REPRESENTATION

1. Extract a weighted low-frequency component from an image.

2. Preserve color composition and surface texture
3. Ignore edges, textures and details
4. it is done by Guided Filtering which uses a differentiable guided filter to extract smooth surface (textures and details removed).

STRUCTURE REPRESENTATION

1. Apply an adaptive coloring algorithm on each segmented regions to generate sparse visual effects.
2. Seize the global structural information
3. Sparse color blocks in celluloid cartoon style

TEXTURE REPRESENTATION

1. Shift the color of the image to generate random intensity maps with luminance and color information removed
2. Retains high-frequency textures
3. Decreases the influence of color and luminance

3.2.4 VALIDATION

The quality of Image cartoonization is highly subjective and greatly influenced by individual preference. But also we evaluate the rate of change in features of the original image and the cartoon.

3.3 Algorithm

Generative adversarial networks (GANs) are algorithmic architectures that use two neural networks, pitting one against the other (thus the "adversarial") in order to generate new, synthetic instances of data that can pass for real data. They are used widely in image generation, video generation and voice generation.

Here are the steps a GAN takes:

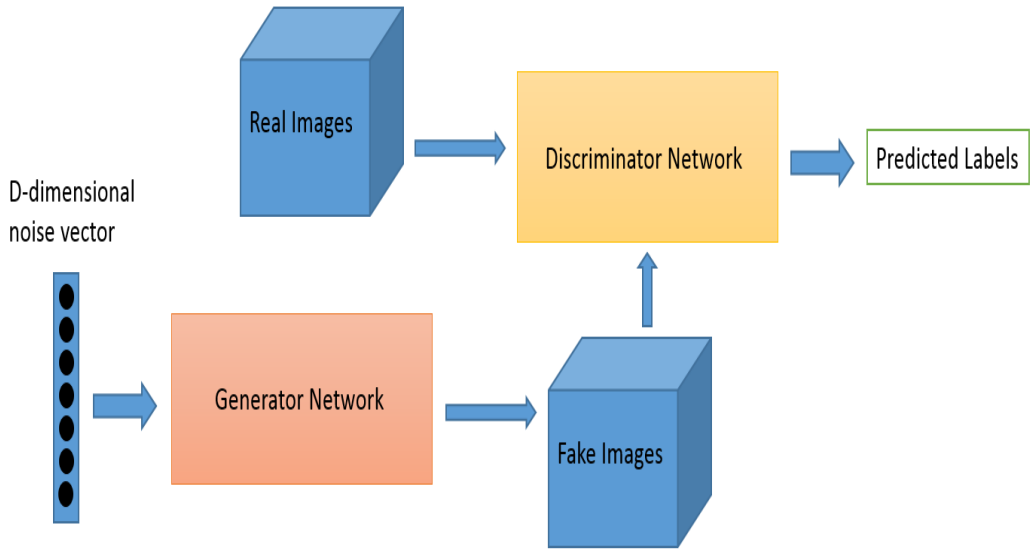
1.The generator takes in random numbers and returns an image.

2. This generated image is fed into the discriminator alongside a stream of images taken from the actual, ground-truth dataset.

3. The discriminator takes in both real and fake images and returns probabilities, a number between 0 and 1, with 1 representing a prediction of authenticity and 0 representing fake.

4.The discriminator is in a feedback loop with the ground truth of the images, which we know.

5. The generator is in a feedback loop with the discriminator.



Chapter 4

RESULT ANALYSIS

The result of the proposed project Image to cartoon conversion using GAN lies in developing a model that can successfully convert a real world image to cartoon image.The proposed system takes a original real world image as input and outputs a cartoon image.

Chapter 5

CONCLUSION AND FUTURE SCOPE

5.1 Conclusion

In this paper we proposed CartoonGAN, a Generative Adversarial Network to transform real-world photos to high-quality cartoon style images

5.2 Future Scope

The project showed that image was successfully converted into a cartoon-style image with help of Cartoon GAN In future this technique can be used to transform video clips into animation clip. For that video should be divided into large number of frames and conversion of each frame should be done

. .

Chapter 6

APPENDIX

6.1 Source code

6.1.1 white_box_catoonization.ipynb

```
import tensorflow
print(tensorflow.__version__)
import os
import cv2
import numpy as np
import tensorflow as tf
import network
import guided_filter
from tqdm import tqdm
from IPython.display import Image

def resize_crop(image):
h, w, c = np.shape(image)
if min(h, w) ¿ 720:
```

```
if h ¿ w:
h, w = int(720*h/w), 720
else:
h, w = 720, int(720*w/h)
image = cv2.resize(image, (w, h),
interpolation=cv2.INTER_AREA)
h, w = (h//8)*8, (w//8)*8
image = image[:h, :w, :]
return image
def cartooniz(load_folder, save_folder, model_path):
input_photo = tf.placeholder(tf.float32, [1, None, None, 3])
network_out = network.unet_generator(input_photo)
final_out = guided_filter.guided_filter(input_photo, network_out, r=1, eps=5e-
3)
all_vars = tf.trainable_variables()
gene_vars = [var for var in all_vars if 'generator' in var.name]
saver = tf.train.Saver(var_list=gene_vars)
config = tf.ConfigProto()
config.gpu_options.allow_growth = True
sess = tf.Session(config=config)
sess.run(tf.global_variables_initializer())
saver.restore(sess, tf.train.latest_checkpoint(model_path))
name_list = os.listdir(load_folder)
for name in tqdm(name_list):
try:

load_path = os.path.join(load_folder, name)
save_path = os.path.join(save_folder, name)
image = cv2.imread(load_path)
image = resize_crop(image)
```

```
batch_image = image.astype(np.float32)/127.5 - 1
batch_image = np.expand_dims(batch_image, axis=0)
output = sess.run(final_out, feed_dict=input_photo: batch_image)
output = (np.squeeze(output)+1)*127.5
output = np.clip(output, 0, 255).astype(np.uint8)
cv2.imwrite(save_path, output)
except:
print('cartoonize  failed'.format(load_path))

model_path = 'saved_models'
load_folder = 'test_images'
save_folder = 'cartoonized_images'
not os.path.exists(save_folder):
os.mkdir(save_folder)
cartooniz(load_folder, save_folder, model_path)
```

6.1.2 guided_filter.py

```
import tensorflow as tf
import numpy as np


def tf_box_filter(x, r):
k_size = int(2*r+1)
ch = x.get_shape().as_list()[-1]
weight = 1/(k_size**2)
box_kernel = weight*np.ones((k_size, k_size, ch, 1))
box_kernel = np.array(box_kernel).astype(np.float32)
output = tf.nn.depthwise_conv2d(x, box_kernel, [1, 1, 1, 1], 'SAME')
return output

def guided_filter(x, y, r, eps=1e-2):

x_shape = tf.shape(x)
y_shape = tf.shape(y)

N = tf_box_filter(tf.ones((1, x_shape[1], x_shape[2], 1), dtype=x.dtype), r)

mean_x = tf_box_filter(x, r) / N
mean_y = tf_box_filter(y, r) / N
cov_xy = tf_box_filter(x * y, r) / N - mean_x * mean_y
var_x = tf_box_filter(x * x, r) / N - mean_x * mean_x

A = cov_xy / (var_x + eps)
b = mean_y - A * mean_x
```

```
mean_A = tf_box_filter(A, r) / N
mean_b = tf_box_filter(b, r) / N

output = mean_A * x + mean_b

return output

def fast_guided_filter(lr_x, lr_y, hr_x, r=1, eps=1e-8):

assert lr_x.shape.ndims == 4 and lr_y.shape.ndims == 4 and hr_x.shape.ndims
== 4

lr_x_shape = tf.shape(lr_x)
lr_y_shape = tf.shape(lr_y)
hr_x_shape = tf.shape(hr_x)

N = tf_box_filter(tf.ones((1, lr_x_shape[1], lr_x_shape[2], 1), dtype=lr_x.dtype),
r)

mean_x = tf_box_filter(lr_x, r) / N
mean_y = tf_box_filter(lr_y, r) / N
cov_xy = tf_box_filter(lr_x * lr_y, r) / N - mean_x * mean_y
var_x = tf_box_filter(lr_x * lr_x, r) / N - mean_x * mean_x

A = cov_xy / (var_x + eps)
b = mean_y - A * mean_x
```

```
mean_A = tf.image.resize_images(A, hr_x_shape[1: 3])
mean_b = tf.image.resize_images(b, hr_x_shape[1: 3])

output = mean_A * hr_x + mean_b

return output

if __name__ == '__main__':
import cv2
from tqdm import tqdm

input_photo = tf.placeholder(tf.float32, [1, None, None, 3])
input_superpixel = tf.placeholder(tf.float32, [16, 256, 256, 3])
output = guided_filter(input_photo, input_photo, 5, eps=1)
image = cv2.imread('output_figure1/cartoon2.jpg')
image = image/127.5 - 1
image = np.expand_dims(image, axis=0)

config = tf.ConfigProto()
config.gpu_options.allow_growth = True
sess = tf.Session(config=config)
sess.run(tf.global_variables_initializer())

out = sess.run(output, feed_dict=input_photo: image)
out = (np.squeeze(out)+1)*127.5
out = np.clip(out, 0, 255).astype(np.uint8)
cv2.imwrite('output_figure1/cartoon2_filter.jpg', out)
```

6.1.3 network.py

```
import tensorflow as tf
import numpy as np
import tensorflow.contrib.slim as slim


def resblock(inputs, out_channel=32, name='resblock'):
with tf.variable_scope(name):
x = slim.convolution2d(inputs, out_channel, [3, 3], activation_fn=None, scope='conv1')
x = tf.nn.leaky_relu(x)
x = slim.convolution2d(x, out_channel, [3, 3], activation_fn=None, scope='conv2')

return x + inputs

def unet_generator(inputs, channel=32, num_blocks=4, name='generator',
reuse=False):
with tf.variable_scope(name, reuse=reuse):

x0 = slim.convolution2d(inputs, channel, [7, 7], activation_fn=None)
x0 = tf.nn.leaky_relu(x0)

x1 = slim.convolution2d(x0, channel, [3, 3], stride=2, activation_fn=None)
x1 = tf.nn.leaky_relu(x1)
x1 = slim.convolution2d(x1, channel*2, [3, 3], activation_fn=None)
x1 = tf.nn.leaky_relu(x1)

x2 = slim.convolution2d(x1, channel*2, [3, 3], stride=2, activation_fn=None)
x2 = tf.nn.leaky_relu(x2)
```

```
x2 = slim.convolution2d(x2, channel*4, [3, 3], activation_fn=None)
x2 = tf.nn.leaky_relu(x2)

for idx in range(num_blocks):
x2 = resblock(x2, out_channel=channel*4, name='block_'.format(idx))

x2 = slim.convolution2d(x2, channel*2, [3, 3], activation_fn=None)
x2 = tf.nn.leaky_relu(x2)

h1, w1 = tf.shape(x2)[1], tf.shape(x2)[2]
x3 = tf.image.resize_bilinear(x2, (h1*2, w1*2))
x3 = slim.convolution2d(x3+x1, channel*2, [3, 3], activation_fn=None)
x3 = tf.nn.leaky_relu(x3)
x3 = slim.convolution2d(x3, channel, [3, 3], activation_fn=None)
x3 = tf.nn.leaky_relu(x3)

h2, w2 = tf.shape(x3)[1], tf.shape(x3)[2]
x4 = tf.image.resize_bilinear(x3, (h2*2, w2*2))
x4 = slim.convolution2d(x4+x0, channel, [3, 3], activation_fn=None)
x4 = tf.nn.leaky_relu(x4)
x4 = slim.convolution2d(x4, 3, [7, 7], activation_fn=None)
return x4

if __name__ == '__main__':


pass
```

6.2 SCREEN SHOTS

Figure 6.1: Home Screen

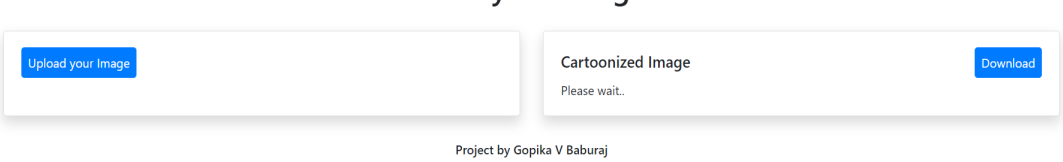


Figure 6.2: Cartoonization

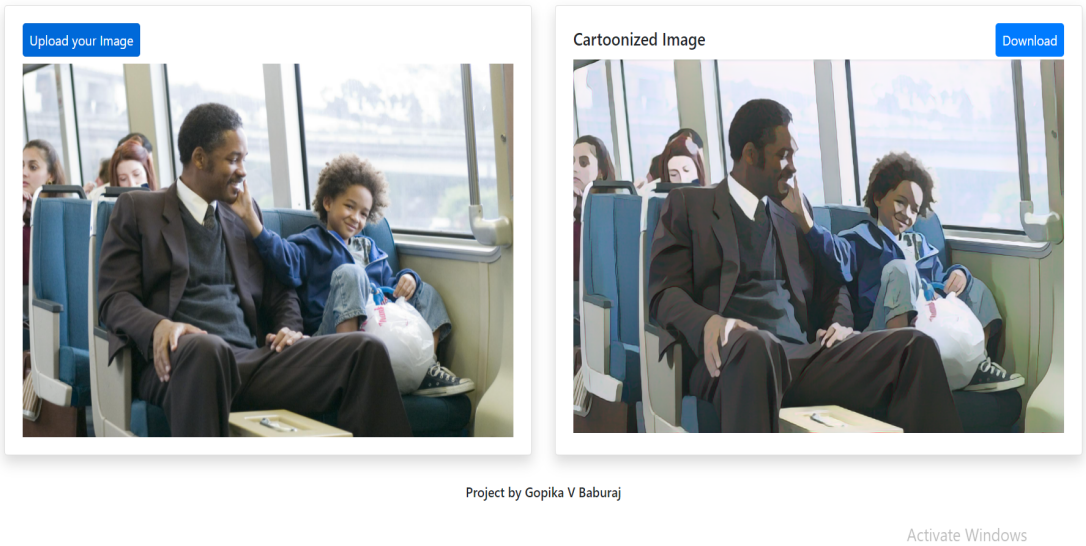


Figure 6.3: Cartoonization

Chapter 7

REFERENCES

1. https://www.researchgate.net/publication/353129069_White-box_Cartoonization_using_an_Extended_GAN_Framework
2. https://towardsai.net/p/deep-learning/an-insiders-guide-to-cartoonization-using-machine-learning
3. https://www.youtube.com/watch?v=eTMGoXgq6uM
4. https://openaccess.thecvf.com/content_cvpr_2018/papers/Chen_CartoonGAN_Generative_Adversarial_CVPR_2018_paper.pdf
5. https://github.com/SystemErrorWang/White-box-Cartoonization