Testing the model on Test sketches and predicting the output image

Group Number : S21DL20

Korupoulu Saideepthi - S20180010087

Varakala Sowmya - S20180010187

Manjju Shree Devy - S20180010055

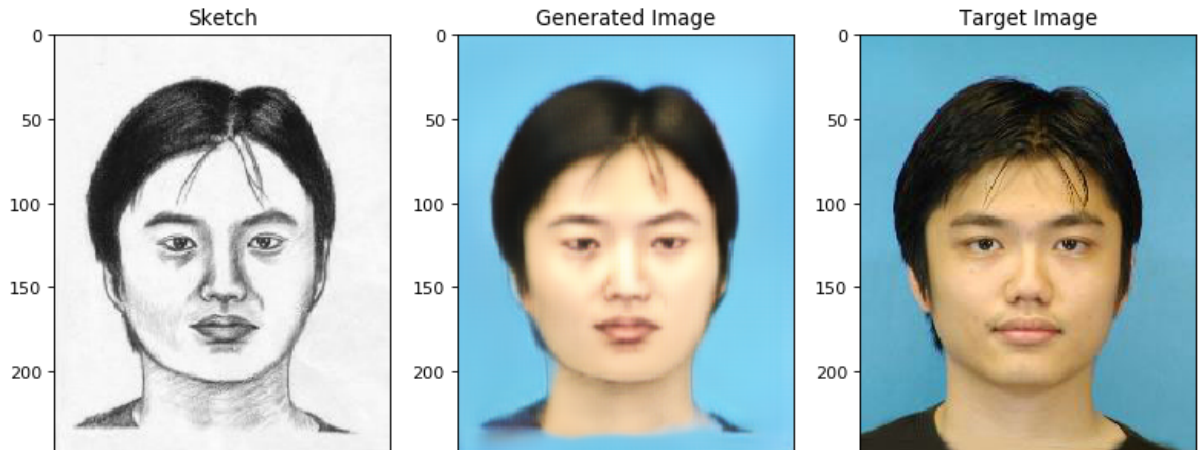
Swathi Kedarasetty - S20180010172

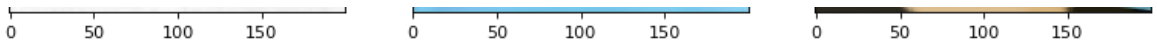
In [1]:
```
# importing necessary libraries
from keras_contrib.layers.normalization.instancenormalization import In
stanceNormalization
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import load_img
from keras.models import load_model
import matplotlib.pyplot as plt
import numpy as np
import cv2
```

```
Using TensorFlow backend.
```

In [3]:
```
# Load Model
g_model = load_model('Models/Pixel[02]_Context[08]/g_model.h5',custom_o
bjects={'InstanceNormalization':InstanceNormalization})

# load and resize the image
img = load_img('Dataset/CUHK/Testing sketch/m1-001-01-sz1.jpg', target_
```

```
size=(256, 256))
target = cv2.cvtColor(cv2.imread('Dataset/CUHK/Testing photo/m1-001-01.
jpg'), cv2.COLOR_BGR2RGB)

# convert to numpy array
img = img_to_array(img)
norm_img = (img.copy() - 127.5) / 127.5

g_img = g_model.predict(np.expand_dims(norm_img, 0))[0]
g_img = g_img * 127.5 + 127.5

img = cv2.resize(img, (200, 250))
g_img = cv2.resize(g_img, (200, 250))

f = plt.figure(num=None, figsize=(12, 6), dpi=80)
ax1 = f.add_subplot(1,3, 1)
plt.imshow(img.astype('uint8'))
ax2 = f.add_subplot(1,3, 2)
plt.imshow(g_img.astype('uint8'))
ax3 = f.add_subplot(1,3, 3)
plt.imshow(target.astype('uint8'))
ax1.set_title('Sketch')
ax2.set_title('Generated Image')
ax3.set_title('Target Image')

plt.show(block=True)
```





In [4]:
```
# importing necessary libraries
from keras_contrib.layers.normalization.instancenormalization import In
stanceNormalization
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import load_img
from keras.models import load_model
import numpy as np
import natsort
import cv2
import os
```

In [5]:
```
def load_filename(path):
    dirFiles = os.listdir(path)
    for i, file in enumerate(dirFiles):
        dirFiles[i] = path + file
    return natsort.natsorted(dirFiles ,reverse=False)

# load all images in a directory into memory
def load_images(list_path, size=(256, 256)):
    img_list = list()
    # enumerate filenames in directory, assume all are images
    for filename in list_path:
        # load and resize the image
        pixels = load_img(filename, target_size=size)
        # convert to numpy array
        pixels = img_to_array(pixels)
        pixels = (pixels - 127.5) / 127.5
        img_list.append(pixels)
    return np.asarray(img_list)

def pred_images(g_model, target_dir, filenames, batch_size=128):
    if not os.path.exists(target_dir):
        os.mkdir(target_dir)

    imgs = load_images(filenames)
    g_img = g_model.predict(imgs)
```

```
    g_img = g_img * 127.5 + 127.5
    for j, _img in enumerate(g_img):
        cv2.imwrite(target_dir + "/" + os.path.basename(filenames[j]),
cv2.resize(cv2.cvtColor(_img.astype('uint8'), cv2.COLOR_RGB2BGR), (200,
250)))
    print("Image has been successfully saved in \"" + target_dir + "\"
 folder")
```

In [6]:

```
# loading the test sketches
filenames = load_filename('Dataset/CUHK/Testing sketch/')
```

In [7]:

```
# importing the model and predicting the images
g_model = load_model('Models/Pixel[02]_Context[08]/g_model.h5',custom_o
bjects={'InstanceNormalization':InstanceNormalization})

pred_images(g_model, "Generated Images/Generated_Pixel[02]_Context[08]"
, filenames)
```

```
WARNING:tensorflow:No training configuration found in the save file, so
the model was *not* compiled. Compile it manually.
Image has been successfully saved in "Generated Images/Generated_Pixel
[02]_Context[08]" folder
```

In [ ]: