TECHNOLOGICAL INSTITUTE OF THE PHILIPPINES
938 Aurora Blvd., Cubao, Quezon City

COLLEGE OF ENGINEERING AND ARCHITECTURE
ELECTRONICS ENGINEERING DEPARTMENT

1ST SEMESTER SY 2022 - 2023

Prediction and Machine Learning

COE 005
ECE41S11

Homework 2
Neural Style Transfer

Submitted to:
Engr. Christian Lian Paulo Rioflorido

Submitted on:
October 18, 2022

Submitted by:
Gian Jemuel C. Laban

HOMEWORK 2: NEURAL STYLE TRANSFER

IMPORT LIBRARIES

```
In [1]: import os
        import tensorflow as tf
        # Load compressed models from tensorflow_hub
        os.environ['TFHUB_MODEL_LOAD_FORMAT'] = 'COMPRESSED'
```

```
In [2]: import IPython.display as display

        import matplotlib.pyplot as plt
        import matplotlib as mpl
        mpl.rcParams['figure.figsize'] = (12, 12)
        mpl.rcParams['axes.grid'] = False

        import numpy as np
        import PIL.Image
        import time
        import functools
```

```
In [3]: def tensor_to_image(tensor):
          tensor = tensor*255
          tensor = np.array(tensor, dtype=np.uint8)
          if np.ndim(tensor)>3:
            assert tensor.shape[0] == 1
            tensor = tensor[0]
          return PIL.Image.fromarray(tensor)
```

Figure 1. Import libraries

First we import the necessary libraries. The os module is one of the standard libraries within python 3, but it is still manually imported. The os module allows us to set a directory path for file extraction of contents to be used as datasets like pictures. Tensorflow is one of the most widely used libraries for deep learning applications in image processing.

LOAD LOCAL DIRECTORY IMAGE DATASET

```
In [11]: #Two image styles and one content
         content_path = "Photo B.jpg"
         style_path = "Ukiyo e.jpg"
         content_path2 = "Photo B.jpg"
         style_path2 = "Fernando Amorsolo.jpg"
```

```
In [12]: #define a function for image resizing
         def load_img(path_to_img):
           max_dim = 512
           img = tf.io.read_file(path_to_img)
           img = tf.image.decode_image(img, channels=3)
           img = tf.image.convert_image_dtype(img, tf.float32)

           shape = tf.cast(tf.shape(img)[:-1], tf.float32)
           long_dim = max(shape)
           scale = max_dim / long_dim

           new_shape = tf.cast(shape * scale, tf.int32)

           img = tf.image.resize(img, new_shape)
           img = img[tf.newaxis, :]
           return img
```

Figure 2: Load dataset

In this figure, a block of code was used for importing the datasets. 4 variables were created to contain the local directory path for the images. Since the image files were in the same folder as the ipynb file containing the code, a simple string with the name of the image file was enough to extract the picture. A function is also defined for image resizing to be used later on.

PHOTO B and UKIYO E

```
In [18]: content_image = load_img(content_path)
         style_image = load_img(style_path)
         content_image2 = load_img(content_path2)
         style_image2 = load_img(style_path2)

         plt.subplot(1, 2, 1)
         imshow(content_image, 'Content Image')

         plt.subplot(1, 2, 2)
         imshow(style_image, 'Style Image')
```

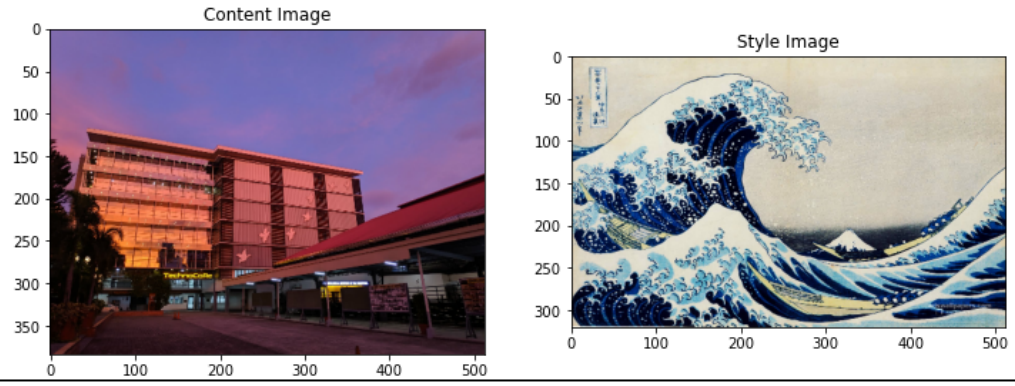


Figure 3: Base image and style image

The dataset images that were used for the first output were the Photo B of TIP as the base image and the famous “The Great Wave off Kanagawa, 1831 by Katsushika Hokusai” as the style image was used for the style transfer. The images were resized to be plotted side by side with a max dimension of 512 pixels. “content_image” variable was used to contain the function “load_img” with the argument being the directory for the image used.

PHOTO B and FERNANDO AMORSOLO ART STYLE

```
In [19]: plt.subplot(2, 2, 3)
         imshow(content_image2, 'Content Image2')

         plt.subplot(2, 2, 4)
         imshow(style_image2, 'Style Image2')
```

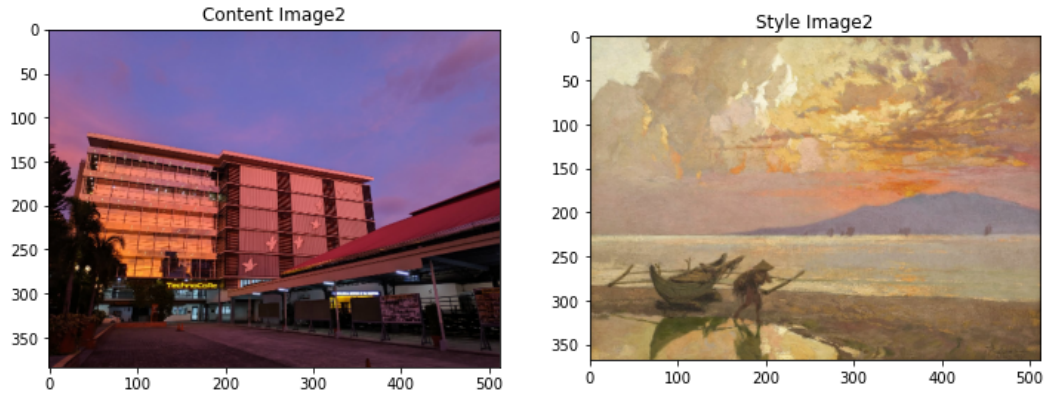


Figure 4: Base image and style image 2

For the second style transfer, the base image is still Photo B of TIP and the style image is “Sunset by the sea” 1935 by Fernando Amorsolo. The same resizing parameters were used for them to be plotted side by side on a smaller scale.

```
import tensorflow_hub as hub
hub_model = hub.load('https://tfhub.dev/google/magenta/arbitrary-image-stylization-v1-256/2')
stylized_image = hub_model(tf.constant(content_image), tf.constant(style_image))[0]
tensor_to_image(stylized_image)
```

Figure 5: Photo B and Ukiyo e fast style transfer

In figure 5, a fast style transfer was conducted using a pre-trained model. The blending of the images were clean and the style of Ukiyo e was apparent in the base image Photo B.

```
hub_model = hub.load('https://tfhub.dev/google/magenta/arbitrary-image-stylization-v1-256/2')
stylized_image2 = hub_model(tf.constant(content_image2), tf.constant(style_image2))[0]
tensor_to_image(stylized_image2)
```

Figure 6: Photo B and Fernando Amorsolo fast style transfer

In figure 6, a fast style transfer was also conducted using a pre-trained model. The blending of the images were clean and the style of Fernando Amorsolo was apparent in the base image Photo B with the nice blend of warm colors.

```
In [ ]: def vgg_layers(layer_names):
          """ Creates a VGG model that returns a list of intermediate output values."""
          # Load our model. Load pretrained VGG, trained on ImageNet data
          vgg = tf.keras.applications.VGG19(include_top=False, weights='imagenet')
          vgg.trainable = False

          outputs = [vgg.get_layer(name).output for name in layer_names]

          model = tf.keras.Model([vgg.input], outputs)
          return model
```

Figure 7: Building the model

The model used was VGG-19 and is built inside a function

```
In [ ]: import time
        start = time.time()

        epochs = 10
        steps_per_epoch = 10

        step = 0
        for n in range(epochs):
          for m in range(steps_per_epoch):
            step += 1
            train_step(image)
            print(".", end='', flush=True)
          display.clear_output(wait=True)
          display.display(tensor_to_image(image))
          print("Train step: {}".format(step))

        end = time.time()
        print("Total time: {:.1f}".format(end-start))
```

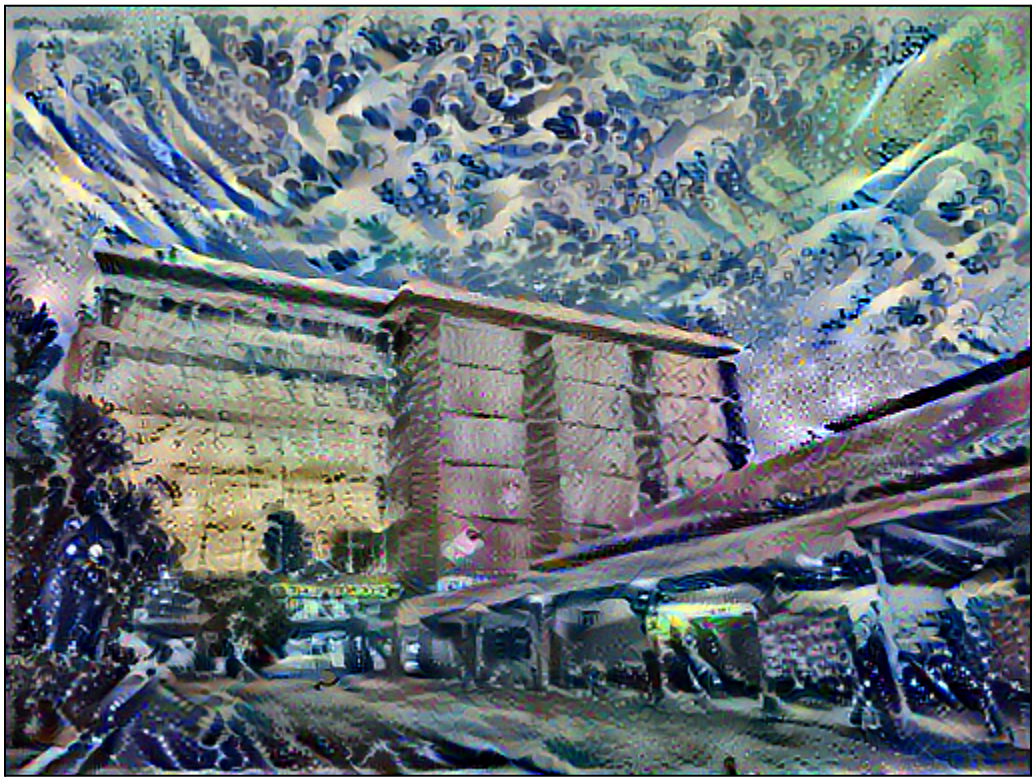
Figure 8: Training parameters for the style transfer

In the figure, epochs will determine the total number of training iterations that will be done. Steps per epoch pertains to the batch size or the number of times the dataset will be evaluated by the network per epoch.

```
Train step: 100
Total time: 225.2
```

Figure 9: image output with 10 epochs and 10 steps per epoch

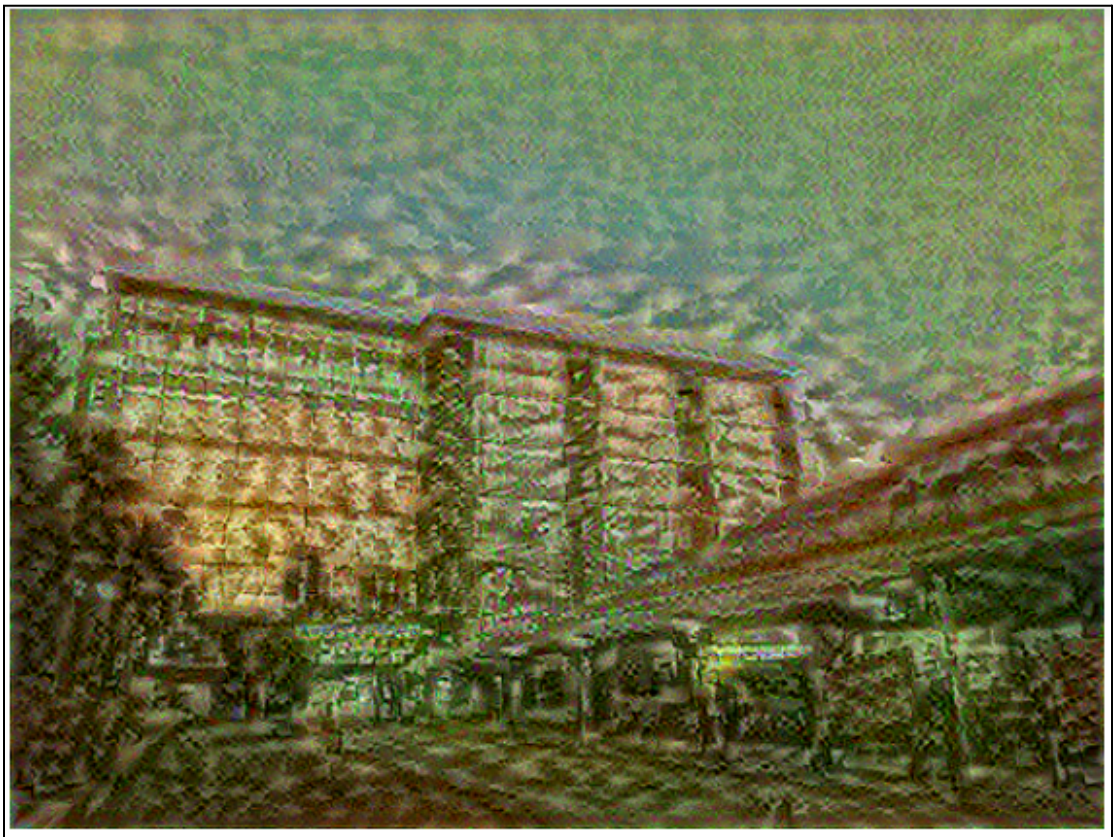
After defining the VGG19 model and adjusting the training parameters, this output of Photo B and Ukiyo e style of the “Great Cost off Kanagawa” was produced using 10 epochs and 10 steps per epoch resulting to 10*10 = 100 train steps. The training time took 225.2 seconds and the image style transfer is not that clean as compared with the pre-trained model.



```
Train step: 500
Total time: 1093.8
```

Figure 6: image output with 10 epochs and 50 steps per epoch

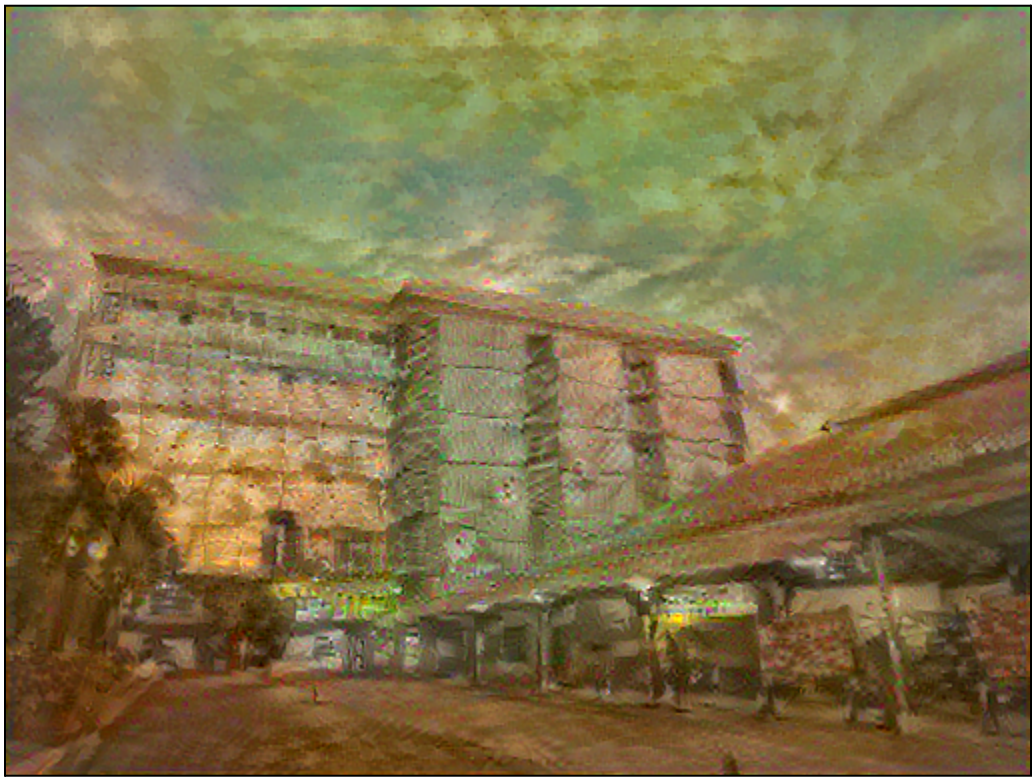
This is the output of Photo B and Ukiyo e art style of the “Great Cost off Kanagawa” that was produced after parameter tuning using 10 epochs and 50 steps per epoch resulting to 10*50 = 500 train steps. The training time took 1093.8 seconds, the image style transfer resulted in a significantly cleaner and refined look as well.



```
Train step: 100
Total time: 214.7
```

Figure 7: image output 2 with 10 epochs and 10 steps per epoch

This output of Photo B and Fernando Amorsolo art style with his artwork “Sunset by the sea” was produced using 10 epochs and 10 steps per epoch resulting to 10*10 = 100 train steps. The training time took 214.7 seconds. The image style transfer is not refined and does not even partially resemble Amorsolo’s art style.



```
Train step: 500
Total time: 1072.1
```

Figure 5: image output 2 with 10 epochs and 50 steps per epoch

This is the output of Photo B and Fernando Amorsolo art style based from his painting of “Sunset by the sea” was produced using 10 epochs and 50 steps per epoch resulting to 10*50 = 500 train steps. Due to the limited hardware, the training time took 1072.1 seconds. The image style transfer is not perfect but it partially resembles Amorsolo’s art style with his significant use of warm colors to portray the sky with the sun setting.