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

GAN Exercise

Submitted to:
Engr. Christian Lian Paulo Rioflorido

Submitted on:
October 21, 2022

Submitted by:
Gian Jemuel C. Laban

GAN Exercise

IMPORT LIBRARIES

```
[7]
import os
import pathlib
import time
import datetime
import imageio
from glob import glob
from google.colab import drive
drive.mount('/content/drive')
os.chdir('/content/drive/My Drive/Colab Notebooks')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
[8]
import tensorflow as tf
import numpy as np
import tensorflow.keras.backend as K
import tensorflow_addons as tfa
```

```
[9]
from matplotlib import pyplot as plt
from IPython import display
from termcolor import colored
from tqdm import tqdm
from IPython.display import Image
import PIL
from PIL import ImageDraw
from IPython import display

def color_print(print_str,
                print_color='green'):

    print(colored(print_str,print_color))
```

Figure 1: Importing of libraries

Using Google colab, importing and mounting the drive makes it possible to store large datasets. The drive creates a special folder specifically for colab notebooks in the same format as Jupyter. In this activity, an image to image translation application of Generative Adversarial Network is used under the Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks or CycleGAN, in other words, it is an unsupervised type of machine learning.

LOAD DATASET

```
[11]
day_dir = 'day'
night_dir = 'night'

#plotting a sample image
plt.figure(figsize=(16,8))

img = plt.imread(day_dir + '/' + os.listdir(day_dir)[0])
plt.imshow(img)
plt.axis('off')
plt.title('sample image')
print(f'Image dimensions {img.shape}')
plt.show()
```

Figure 2: Load the dataset

The dataset that was used for this exercise are pictures of night and day cityscapes. The purpose of the notebook is to train a model on image to image translation as one of the applications of generative adversarial networks. In the figure above, the directory was simply a string name of the folders used as the base image dataset “day” and the target image dataset “night” since I already changed the directory to be used by colab through the import os function and setting it to the folder covering the dataset folders.

```
Day_Dataset = create_img_dataset(directory = day_dir,image_preprocess_fn = preprocess_image_train)

#without augmentation
Day_eval = create_img_dataset(directory = day_dir, image_preprocess_fn = preprocess_image_eval)

fig,ax = plt.subplots(figsize=(16,8))

inp_img = next(iter(Day_Dataset))
plt.imshow(de_normalize(inp_img[0]))
plt.title('Sample Day image')
plt.axis('off')

plt.show()
```

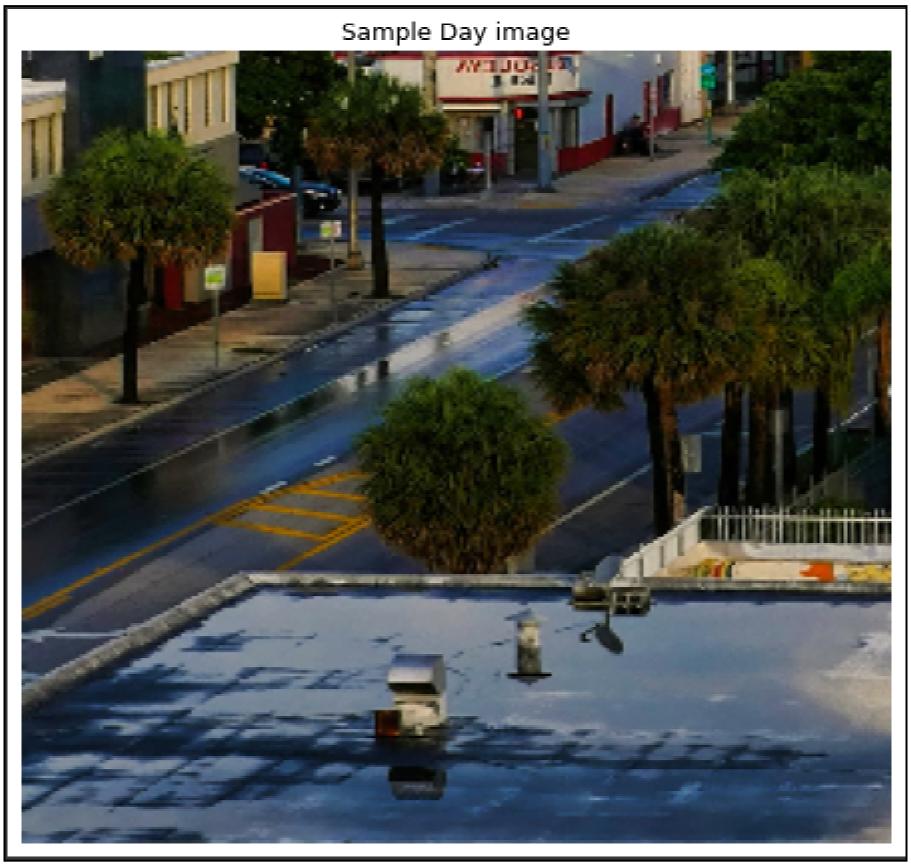

Figure 3: Day image sample.

The block of code above is to get a sample of a day image from the dataset. A day image is used to train the model to make the image to image translation. The day image dataset consists of pictures of cities with clear backgrounds and the sky illuminated by the sun on morning to afternoon where the sun shines its brightest projections of light.

```
Night_Dataset = create_img_dataset(directory = night_dir,image_preprocess_fn = preprocess_image_train)
Night_eval = create_img_dataset(directory = night_dir,image_preprocess_fn = preprocess_image_eval)
fig,ax = plt.subplots(figsize=(16,8))

inp_img = next(iter(Night_Dataset))
plt.imshow(de_normalize(inp_img[0]))
plt.title('Sample Night image')
plt.axis('off')

plt.show()
```

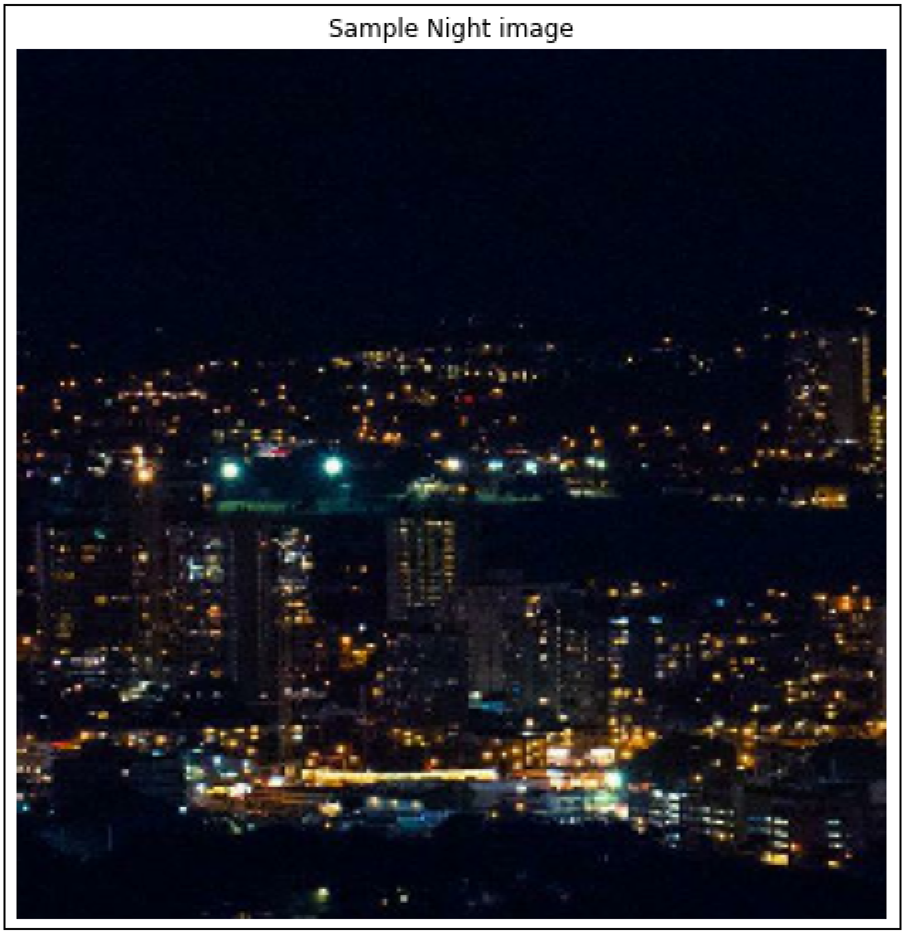

Figure 4: Importing of dataset

From the figure above, a sample of a night image is also taken from the directory and is shown as a snippet of what the entire night images as part of the training dataset looks like. The night image dataset consists of pictures of cities with the sky in almost complete absence of light for high contrast between the source and the target domains for the discriminator network to have an easier time telling what is real and what is fake data from the generator later.

```
[ ]  EPOCHS = 80
     callbacks = [lr_scheduler,GANMonitor(),CustomEarlyStopping(patience = 10)]
     steps_per_epoch = 200

     history = gan.fit(Train_Dataset,epochs = EPOCHS,steps_per_epoch=steps_per_epoch,callbacks = callbacks)

Epoch 1/80
200/200 [============================] - 894s 4s/step - gen_D2N_loss: 9.4679 - gen_N2D_loss: 9.4037 - disc_day_loss: 0.6010 - disc_night_loss: 0.6132
Epoch 2/80
200/200 [============================] - 850s 4s/step - gen_D2N_loss: 8.8211 - gen_N2D_loss: 8.6886 - disc_day_loss: 0.6446 - disc_night_loss: 0.6168
Epoch 3/80
200/200 [============================] - 848s 4s/step - gen_D2N_loss: 8.5689 - gen_N2D_loss: 8.4729 - disc_day_loss: 0.6506 - disc_night_loss: 0.6831
Epoch 4/80
200/200 [============================] - 851s 4s/step - gen_D2N_loss: 8.4782 - gen_N2D_loss: 8.4026 - disc_day_loss: 0.5642 - disc_night_loss: 0.6792
Epoch 5/80
200/200 [============================] - 846s 4s/step - gen_D2N_loss: 8.4714 - gen_N2D_loss: 8.2141 - disc_day_loss: 0.6456 - disc_night_loss: 0.6739
Epoch 6/80
200/200 [============================] - 847s 4s/step - gen_D2N_loss: 8.3602 - gen_N2D_loss: 7.8944 - disc_day_loss: 0.6682 - disc_night_loss: 0.6837
Epoch 7/80
200/200 [============================] - 847s 4s/step - gen_D2N_loss: 8.2750 - gen_N2D_loss: 7.8081 - disc_day_loss: 0.6481 - disc_night_loss: 0.6640
Epoch 8/80
200/200 [============================] - 845s 4s/step - gen_D2N_loss: 8.3414 - gen_N2D_loss: 7.6903 - disc_day_loss: 0.6490 - disc_night_loss: 0.6575
Epoch 9/80
200/200 [============================] - 845s 4s/step - gen_D2N_loss: 8.4592 - gen_N2D_loss: 7.6381 - disc_day_loss: 0.6636 - disc_night_loss: 0.6325
Epoch 10/80
200/200 [============================] - 847s 4s/step - gen_D2N_loss: 8.5938 - gen_N2D_loss: 7.7445 - disc_day_loss: 0.6462 - disc_night_loss: 0.5796
```

Figure 5: Training the model

For the model training of the CycleGAN, an epoch of 80 and steps per epoch of 200 was used as seen from the figure above. Early stopping was also used to prevent overfitting. A Generative Adversarial Network has two parts, a Generator and a Discriminator . The Generator creates a random sample that the Discriminator evaluates on the basis of using the real images from the datasets as samples as seen in the samples of both the day and night datasets.The Generator network is the one that undergoes a training that upon success, can generate an image that can fool the Discriminator from telling the real data from the generated fake data, hence an accurate image translation that is close to the target domain, which in this case, the night images.

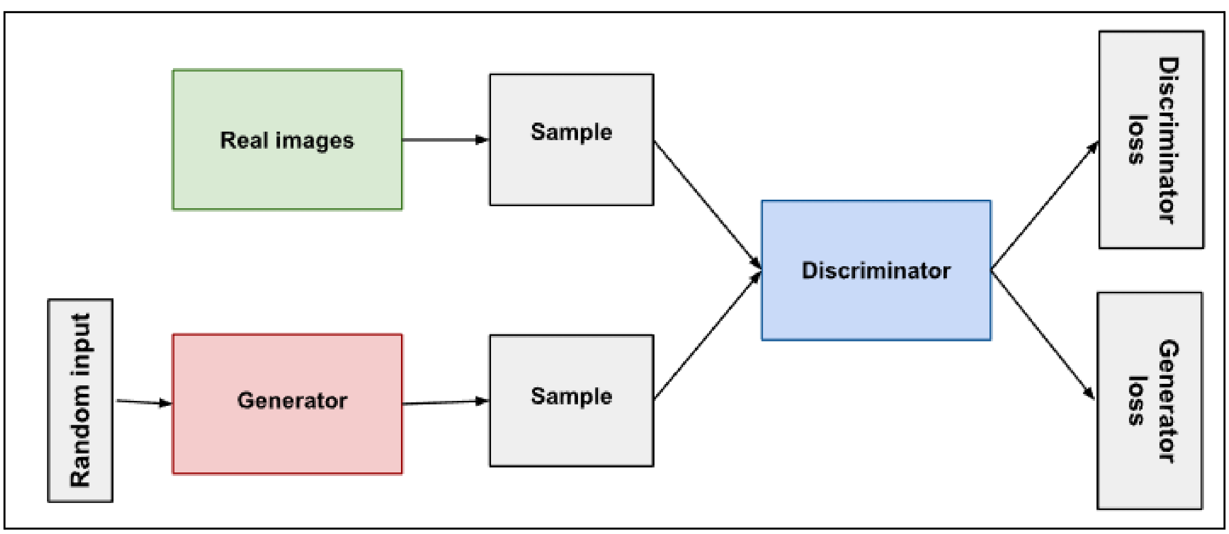

Figure 6: GAN structure (source: https://developers.google.com/)

```
1 evaluate_cycle(Day_eval.take(2), day2night_gen, night2day_gen, n_samples=2)
```

Figure 7: Day to night and night to day image translation

Here, since the base image is a day image, and the first generated image is a transition from day to night, and followed from night to day. The unrefined nature of the translated images was due to image resizing of a lower resolution of 600 x 350 pixels. The training was also set to 80 epochs and 200 steps per epoch but was stopped at around the 19th epoch by the early stopping condition.

```
1 evaluate_cycle(Day_eval.take(2), day2night_gen, night2day_gen, n_samples=2)
```

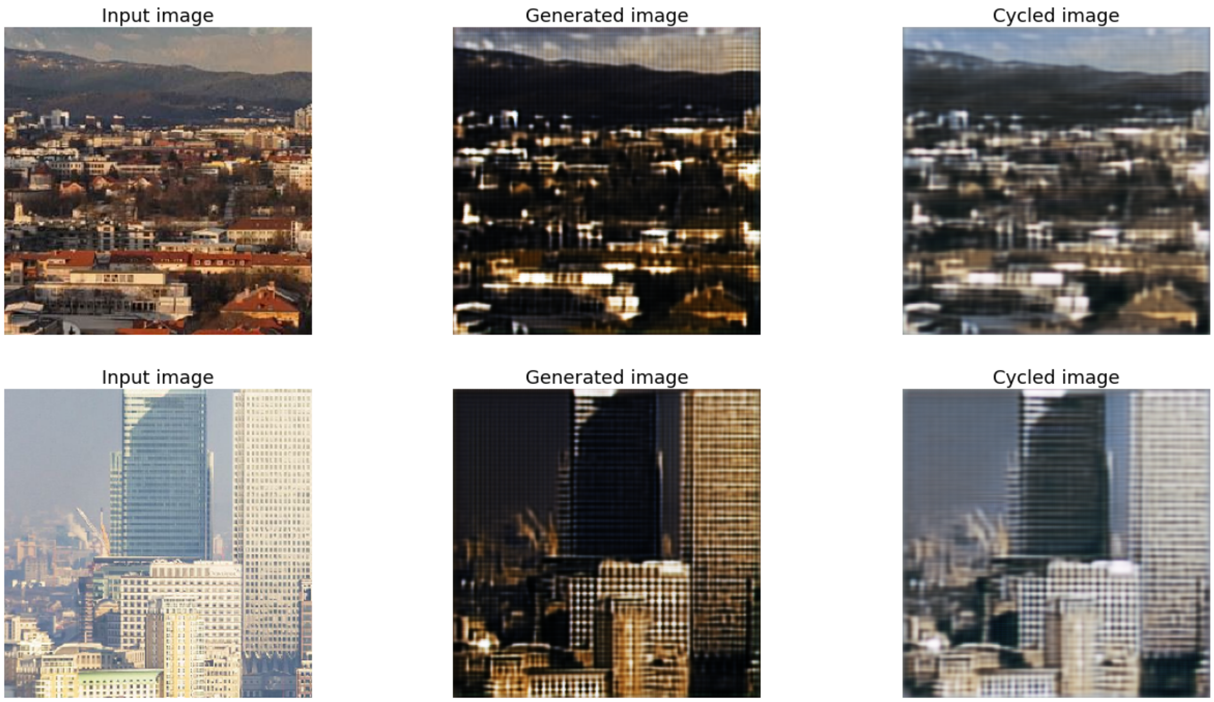

Figure 8: Day to night and night to day image translation

One of the applications of Generative Adversarial Networks (GAN) is image-to-image translation. Cycle GAN is used as a way to generate the image translations in an unsupervised manner, or without the need to manually map the test to train domains; just simply providing the base source image and the target image dataset is enough, the unsupervised model itself will make the translation by itself and will depend on proper training for the best results.

Reference:
https://www.tensorflow.org/tutorials/generative/cyclegan
https://developers.google.com/machine-learning/gan/gan_structure