```
!pip install plot_utils
```

```
Requirement already satisfied: plot_utils in /usr/local/lib/python3.10/dist-packages (0.6.14)
Requirement already satisfied: numpy>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from plot_utils) (1.25.2)
Requirement already satisfied: scipy>=0.19.0 in /usr/local/lib/python3.10/dist-packages (from plot_utils) (1.11.4)
Requirement already satisfied: pandas>=0.17.1 in /usr/local/lib/python3.10/dist-packages (from plot_utils) (1.5.3)
Requirement already satisfied: cycler>=0.10.0 in /usr/local/lib/python3.10/dist-packages (from plot_utils) (0.12.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from plot_utils) (3.7.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.17.1->plot_utils) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.17.1->plot_utils) (2023.4)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->plot_utils) (1.2.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->plot_utils) (4.49.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->plot_utils) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->plot_utils) (23.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->plot_utils) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->plot_utils) (3.1.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas>=0.17.1->plot_utils) (1.16.0)
```

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential, save_model, load_model
from tensorflow.keras.layers import Dense, Conv2DTranspose, Conv2D, Flatten
from tensorflow.keras.layers import Reshape, Dropout, BatchNormalization, LeakyReLU
import numpy as np
import plot_utils
import matplotlib.pyplot as plt
from tqdm import tqdm
from IPython import display
%matplotlib inline
```

```
#loading fashion mnist data from keras
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.fashion_mnist.load_data()

x_train = x_train.astype(np.float32) / 255.0
x_test = x_test.astype(np.float32) / 255.0
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 [==============================] - 3s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 [==============================] - 1s 0us/step
```

```
#using the tensorflow.data.Dataset API to make the training dataset
#we shuffle with buffer size 1000
batch_size = 32
dataset = tf.data.Dataset.from_tensor_slices(x_train).shuffle(1000)
dataset = dataset.batch(batch_size, drop_remainder=True).prefetch(1)
```

```
num_features = 100 #number of dimensions of the input latent variable space(noise)

#Generator
generator = Sequential()
generator.add(Dense(7*7*128, input_shape=(num_features,)))
generator.add(Reshape([7, 7, 128]))
generator.add(BatchNormalization())
generator.add(Conv2DTranspose(64, (5,5), strides = (2,2), padding='same', activation='selu'))
generator.add(BatchNormalization())
generator.add(Conv2DTranspose(1, (5,5), strides = (2,2), padding='same', activation='tanh'))
```

```
#discriminator
discriminator = Sequential()
discriminator.add(Conv2D(64, (5,5), strides = (2,2), padding='same', input_shape=(28,28,1,)))
discriminator.add(LeakyReLU(0.2))
discriminator.add(Dropout(0.3))
discriminator.add(Conv2D(128, (5,5), strides=(2,2), padding='same'))
discriminator.add(LeakyReLU(0.2))
discriminator.add(Dropout(0.3))
discriminator.add(Flatten())
discriminator.add(Dense(1, activation='sigmoid'))
```

```
discriminator.compile(loss='binary_crossentropy', optimizer='rmsprop')
discriminator.trainable = False

#combining the generator and the discriminator
gan = Sequential()
gan.add(generator)
gan.add(discriminator)
gan.compile(loss='binary_crossentropy', optimizer='rmsprop')
```

```
seed = tf.random.normal(shape=(batch_size, num_features))
```

```
def train_dcgan(gan, dataset, batch_size, num_features, epochs=5):
    generator, discriminator = gan.layers
    for epoch in tqdm(range(epochs)):
        print("Epochs {}/{}".format(epoch+1, epochs))
        for x_batch in dataset:
            noise = tf.random.normal(shape=(batch_size, num_features))
            generated_images = generator(noise)
            x_fake_and_real = tf.concat([generated_images, x_batch], axis=0) #input for discriminator
            y1 = tf.constant([[0.]] * batch_size + [[1.]] * batch_size) #output for discriminator
            discriminator.trainable = True
            discriminator.train_on_batch(x_fake_and_real, y1)
            y2 = tf.constant([[1.]] * batch_size)
            discriminator.trainable = False
            gan.train_on_batch(noise, y2)
        display.clear_output(wait=True)
        generate_and_save_images(generator, epoch+1, seed)
    display.clear_output(wait=True)
    generate_and_save_images(generator, epochs, seed)
```

```
def generate_and_save_images(model, epoch, test_input):
    predictions = model(test_input, training=False)
    fig = plt.figure(figsize=(10,10))

    for i in range(25):
        plt.subplot(5, 5, i+1)
        plt.imshow(predictions[i, :, :, 0] * 127.5 + 127.5, cmap='binary') #rescaling the pixel values back to 0-255 range
        plt.axis('off')
    plt.savefig('image_at_epoch_{:04d}.png'.format(epoch))
    plt.show()
```

```
x_train_dc_gan = x_train.reshape(-1, 28, 28, 1) * 2 - 1
```

```
batch_size = 32
dataset = tf.data.Dataset.from_tensor_slices(x_train_dc_gan).shuffle(1000)
dataset = dataset.batch(batch_size, drop_remainder = True).prefetch(1)
```

```
%%time
train_dcgan(gan, dataset, batch_size, num_features, epochs=10)
```

```
CPU times: user 9min 47s, sys: 22.6 s, total: 10min 10s
Wall time: 10min 31s
```