- Name: Rishabh Patil
- SAP: 60009200056 (K/K2)

```
from numpy import zeros, ones, expand_dims, asarray
from numpy.random import randn, randint
from keras.datasets import fashion_mnist
from keras.optimizers import Adam
from keras.models import Model, load_model
from keras.layers import Input, Dense, Reshape, Flatten
from keras.layers import Conv2D, Conv2DTranspose, Concatenate
from keras.layers import LeakyReLU, Dropout, Embedding
from keras.layers import BatchNormalization, Activation
from keras import initializers
from keras.initializers import RandomNormal
from keras.optimizers import Adam, RMSprop, SGD
from matplotlib import pyplot
import numpy as np
from math import sqrt


(X_train, _), (_, _) = fashion_mnist.load_data()
X_train = X_train.astype(np.float32) / 127.5 - 1
X_train = np.expand_dims(X_train, axis=3)
print(X_train.shape)
```

```
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 [==============================] - 0s 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 [==============================] - 0s 0us/step
(60000, 28, 28, 1)
```

```
def define_discriminator(in_shape=(28, 28, 1)):
    #init = RandomNormal(stddev=0.02)
    in_image = Input(shape=in_shape)
    fe = Flatten()(in_image)
    fe = Dense(1024)(fe)
    fe = LeakyReLU(alpha=0.2)(fe)
    fe = Dropout(0.3)(fe)
    fe = Dense(512)(fe)
    fe = LeakyReLU(alpha=0.2)(fe)
    fe = Dropout(0.3)(fe)
    fe = Dense(256)(fe)
    fe = LeakyReLU(alpha=0.2)(fe)
```

```
    fe = Dropout(0.3)(fe)

    out = Dense(1, activation='sigmoid')(fe)

    model = Model(in_image, out)

    opt = Adam(lr=0.0002, beta_1=0.5)

    model.compile(loss='binary_crossentropy', optimizer=opt, metrics=['accuracy'])

    return model

discriminator = define_discriminator()

def define_generator(latent_dim):

    init  = RandomNormal(stddev=0.02) #generates an array of specified shapes and fills it with random values, which is actually a part c

    in_lat = Input(shape=(latent_dim,))

    gen = Dense(256, kernel_initializer=init)(in_lat)

    gen = LeakyReLU(alpha=0.2)(gen)

    gen = Dense(512, kernel_initializer=init)(gen)

    gen = LeakyReLU(alpha=0.2)(gen)

    gen = Dense(1024, kernel_initializer=init)(gen)

    gen = LeakyReLU(alpha=0.2)(gen)

    gen = Dense(28 * 28 * 1, kernel_initializer=init)(gen)

    out_layer = Activation('tanh')(gen)

    out_layer = Reshape((28, 28, 1))(gen)

    model = Model(in_lat, out_layer)

    return model

generator = define_generator(100)

def define_gan(g_model, d_model):

    d_model.trainable = False

    gan_output = d_model(g_model.output)

    model = Model(g_model.input, gan_output)

    opt = Adam(lr=0.0002, beta_1=0.5)

    model.compile(loss='binary_crossentropy', optimizer=opt, metrics=['accuracy'])

    return model

gan_model = define_gan(generator, discriminator)
```

```
/usr/local/lib/python3.8/dist-packages/keras/optimizers/optimizer_v2/adam.py:110: UserWarning: The `lr` argument is deprecated, use
  super(Adam, self).__init__(name, **kwargs)
```

```
def generate_latent_points(latent_dim, n_samples):

    x_input = randn(latent_dim * n_samples)

    z_input = x_input.reshape(n_samples, latent_dim)

    return z_input

def generate_real_samples(X_train, n_samples):

    ix = randint(0, X_train.shape[0], n_samples)

    X = X_train[ix]

    y = ones((n_samples, 1))
```

```
    return X, y

def generate_fake_samples(generator, latent_dim, n_samples):

    z_input = generate_latent_points(latent_dim, n_samples)

    images = generator.predict(z_input)

    y = zeros((n_samples, 1))

    return images, y


def summarize_performance(step, g_model, latent_dim, n_samples=100):

    X, _ = generate_fake_samples(g_model, latent_dim, n_samples)

    X = (X + 1) / 2.0

    for i in range(100):

        pyplot.subplot(10, 10, 1 + i)

        pyplot.axis('off')

        pyplot.imshow(X[i, :, :, 0], cmap='gray_r')

    filename2 = 'model_%04d.h5' % (step+1)

    g_model.save(filename2)

    print('>Saved: %s' % (filename2))

def save_plot(examples, n_examples):

    for i in range(n_examples):

        pyplot.subplot(sqrt(n_examples), sqrt(n_examples), 1 + i)

        pyplot.axis('off')

        pyplot.imshow(examples[i, :, :, 0], cmap='gray_r')

    pyplot.show()


def train(g_model, d_model, gan_model, X_train, latent_dim, n_epochs=100, n_batch=64):

    bat_per_epo = int(X_train.shape[0] / n_batch)

    n_steps = bat_per_epo * n_epochs

    for i in range(n_steps):

        X_real, y_real = generate_real_samples(X_train, n_batch)

        d_loss_r, d_acc_r = d_model.train_on_batch(X_real, y_real)

        X_fake, y_fake = generate_fake_samples(g_model, latent_dim, n_batch)

        d_loss_f, d_acc_f = d_model.train_on_batch(X_fake, y_fake)

        z_input = generate_latent_points(latent_dim, n_batch)

        y_gan = ones((n_batch, 1))

        g_loss, g_acc = gan_model.train_on_batch(z_input, y_gan)

        print('>%d, dr[%.3f,%.3f], df[%.3f,%.3f], g[%.3f,%.3f]' % (i+1, d_loss_r,d_acc_r, d_loss_f,d_acc_f, g_loss,g_acc))

        if (i+1) % (bat_per_epo * 1) == 0:

            summarize_performance(i, g_model, latent_dim)

latent_dim = 100

train(generator, discriminator, gan_model, X_train, latent_dim, n_epochs=20, n_batch=64)

model = load_model('model_18740.h5')

latent_dim = 100
```

```
n_examples = 100

latent_points = generate_latent_points(latent_dim, n_examples)

X  = model.predict(latent_points)

X = (X + 1) / 2.0

save_plot(X, n_examples)
```

```
2/2 [==============================] - 0s 50ms/step
>1, dr[0.957,0.328], df[0.700,0.000], g[0.687,0.984]
2/2 [==============================] - 0s 17ms/step
>2, dr[0.119,1.000], df[0.713,0.000], g[0.675,1.000]
2/2 [==============================] - 0s 16ms/step
>3, dr[0.048,1.000], df[0.726,0.000], g[0.666,1.000]
2/2 [==============================] - 0s 36ms/step
>4, dr[0.024,1.000], df[0.745,0.000], g[0.651,1.000]
2/2 [==============================] - 0s 28ms/step
>5, dr[0.015,1.000], df[0.761,0.000], g[0.636,1.000]
2/2 [==============================] - 0s 22ms/step
>6, dr[0.012,1.000], df[0.793,0.000], g[0.614,1.000]
2/2 [==============================] - 0s 18ms/step
>7, dr[0.013,1.000], df[0.818,0.000], g[0.600,1.000]
2/2 [==============================] - 0s 30ms/step
>8, dr[0.011,1.000], df[0.843,0.000], g[0.595,1.000]
2/2 [==============================] - 0s 21ms/step
>9, dr[0.021,1.000], df[0.858,0.000], g[0.592,1.000]
2/2 [==============================] - 0s 21ms/step
>10, dr[0.016,1.000], df[0.847,0.000], g[0.618,1.000]
2/2 [==============================] - 0s 17ms/step
>11, dr[0.031,1.000], df[0.812,0.000], g[0.659,0.891]
2/2 [==============================] - 0s 16ms/step
>12, dr[0.024,1.000], df[0.756,0.031], g[0.709,0.359]
2/2 [==============================] - 0s 13ms/step
>13, dr[0.023,1.000], df[0.707,0.391], g[0.772,0.016]
2/2 [==============================] - 0s 21ms/step
>14, dr[0.043,1.000], df[0.674,0.734], g[0.829,0.000]
2/2 [==============================] - 0s 35ms/step
>15, dr[0.036,1.000], df[0.660,0.766], g[0.837,0.000]
2/2 [==============================] - 0s 15ms/step
>16, dr[0.028,1.000], df[0.656,0.797], g[0.837,0.000]
2/2 [==============================] - 0s 31ms/step
>17, dr[0.025,1.000], df[0.644,0.906], g[0.845,0.000]
2/2 [==============================] - 0s 23ms/step
>18, dr[0.039,1.000], df[0.666,0.812], g[0.807,0.000]
2/2 [==============================] - 0s 12ms/step
>19, dr[0.040,1.000], df[0.691,0.594], g[0.797,0.000]
2/2 [==============================] - 0s 21ms/step
>20, dr[0.024,1.000], df[0.688,0.547], g[0.790,0.016]
2/2 [==============================] - 0s 15ms/step
>21, dr[0.031,1.000], df[0.692,0.578], g[0.795,0.000]
2/2 [==============================] - 0s 12ms/step
>22, dr[0.023,1.000], df[0.680,0.688], g[0.802,0.000]
2/2 [==============================] - 0s 17ms/step
>23, dr[0.026,1.000], df[0.684,0.609], g[0.820,0.000]
2/2 [==============================] - 0s 11ms/step
>24, dr[0.028,1.000], df[0.663,0.828], g[0.826,0.000]
2/2 [==============================] - 0s 14ms/step
>25, dr[0.021,1.000], df[0.663,0.781], g[0.831,0.000]
2/2 [==============================] - 0s 16ms/step
>26, dr[0.040,0.984], df[0.681,0.594], g[0.823,0.000]
2/2 [==============================] - 0s 19ms/step
>27, dr[0.010,1.000], df[0.673,0.734], g[0.844,0.000]
2/2 [==============================] - 0s 21ms/step
>28, dr[0.017,1.000], df[0.659,0.797], g[0.853,0.000]
2/2 [==============================] - 0s 12ms/step
```