import matplotlib.pyplot as plt

import numpy as np

import pandas as pd #reading

import tensorflow as tf

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score

from sklearn.model\_selection import train\_test\_split #model Selection and Scaling the data

from tensorflow.keras import layers, losses

from tensorflow.keras.datasets import fashion\_mnist

from tensorflow.keras.models import Model

(x\_train, \_), (x\_test, \_)= fashion\_mnist.load\_data()

x\_train=x\_train/255

x\_test=x\_test/255

print(x\_train.shape)

print(x\_test.shape)

latent\_dim=64

class Autoencoder(Model):

def \_\_init\_\_(self, latent\_dim):

super(Autoencoder, self).\_\_init\_\_()

self.latent\_dim=latent\_dim

self.encoder=tf.keras.Sequential([

layers.Flatten(),

layers,Dense(latent\_dim, activation='relu'),

])

self.decoder=tf.keras.Sequential([

layers.Dense(784,activation='sigmoid'),

layers.Reshape((28,28))

])

def call(self, x):

encoded=self.encoder(x)

decoded=self.decoder(encoded)

return decoded

autoencoder=Autoencoder(latent\_dim)

autoencoder.compile(optimizer='adam', loss=losses.MeanSquaredError())

autoencoder.fit(x\_train, x\_train,

epochs=10,

shuffle=True,

validation\_data=(x\_test, x\_test))

encoded\_imgs=autoencoder.encoder(x\_test).numpy()

decoded\_imgs=autoencoder.decoder(encoded\_imgs).numpy()

n=10

plt.figure(figsize=(20,4))

for i in range(n):

#Display original

ax=plt.subplot(2, n,i+1)

plt.imshow(x\_test[i])

plt.title("Original")

plt.gray()

ax.get\_xaxis().set\_visible(False)

ax.get\_yaxis().set\_visible(False)

#Display reconstruction

ax=plt.subplot(2,n,i+1+n)

plt.imshow(decoded\_imgs[i])

plt.title("Reconstructed")

plt.gray()

ax.get\_xaxis().set\_visible(False)

ax.get\_yaxis().set\_visible(False)

plt.show()