Assignment No : 8



import tensorflow as tf import matplotlib.pyplot as plt from tensorflow import keras

import numpy as np

(x\_train, y\_train), (x\_test, y\_test) = keras.datasets.fashion\_mnist.load\_data()

*# There are 10 image classes in this dataset and each class has a mapping corresponding to the following labels: #0 T-shirt/top*

*#1 Trouser*

*#2 pullover*

*#3 Dress*

*#4 Coat*

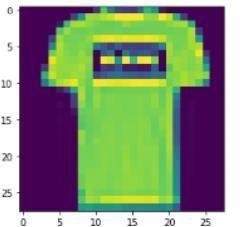
*#5 sandals*

*#6 shirt*

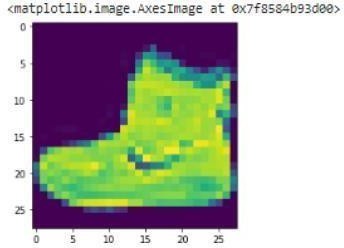
*#7 sneaker #8 bag*

*#9 ankle boot*

plt.imshow(x\_train[1]

)

plt.imshow(x\_train[0])



# Next, we will preprocess the data by scaling the pixel values to be between 0 and 1, and then reshaping the images to be 28x28 pixels. x\_train = x\_train.astype('float32') / 255.0 x\_test = x\_test.astype('float32') / 255.0 x\_train =

x\_train.reshape(-1, 28, 28, 1) x\_test = x\_test.reshape(-1,

28, 28, 1)



# 28, 28 comes from width, height, 1 comes from the number of channels # -1 means that the length in that dimension is inferred.

# This is done based on the constraint that the number of elements in an ndarray or Tensor when reshaped must remain the same.

# each image is a row vector (784 elements) and there are lots of such rows (let it be n, so there are 784n elements). So TensorFlow can infer that -1 is n.

# converting the training\_images array to 4 dimensional array with sizes 60000, 28, 28, 1 for 0th to 3rd dimension. x\_train.shape (60000, 28, 28) x\_test.shape (10000, 28, 28, 1) y\_train.shape (60000,)

y\_test.shape (10000,)

# We will use a convolutional neural network (CNN) to classify the fashion items. # The CNN will consist of multiple convolutional layers followed by max pooling, # dropout, and dense layers. Here is the code for the model: model = keras.Sequential([

keras.layers.Conv2D(32, (3,3), activation='relu', input\_shape=(28,28,1)), # 32 filters (default), randomly initialized

# 3\*3 is Size of Filter

# 28,28,1 size of Input Image

# No zero-padding: every output 2 pixels less in every dimension

# in Paramter shwon 320 is value of weights: (3x3 filter weights + 32 bias) \* 32 filters # 32\*3\*3=288(Total)+32(bias)= 320 keras.layers.MaxPooling2D((2,2)),

# It shown 13 \* 13 size image with 32 channel or filter or depth. keras.layers.Dropout(0.25), # Reduce Overfitting of Training sample drop out 25% Neuron

keras.layers.Conv2D(64, (3,3), activation='relu'), # Deeper layers use 64 filters

# 3\*3 is Size of Filter

# Observe how the input image on 28x28x1 is transformed to a 3x3x64 feature map

# 13(Size)-3(Filter Size )+1(bias)=11 Size for Width and Height with 64 Depth or filtter or channel # in Paramter shwon 18496 is value of weights: (3x3 filter weights + 64 bias) \* 64 filters

# 64\*3\*3=576+1=577\*32 + 32(bias)=18496 keras.layers.MaxPooling2D((2,2)),

# It shown 5 \* 5 size image with 64 channel or filter or depth. keras.layers.Dropout(0.25), keras.layers.Conv2D(128, (3,3), activation='relu'),

# Deeper layers use 128 filters # 3\*3 is Size of Filter

# Observe how the input image on 28x28x1 is transformed to a 3x3x128 feature map # It show 5(Size)-3(Filter Size )+1(bias)=3 Size for Width and Height with 64 Depth or filtter or channel

# 128\*3\*3=1152+1=1153\*64 + 64(bias)= 73856

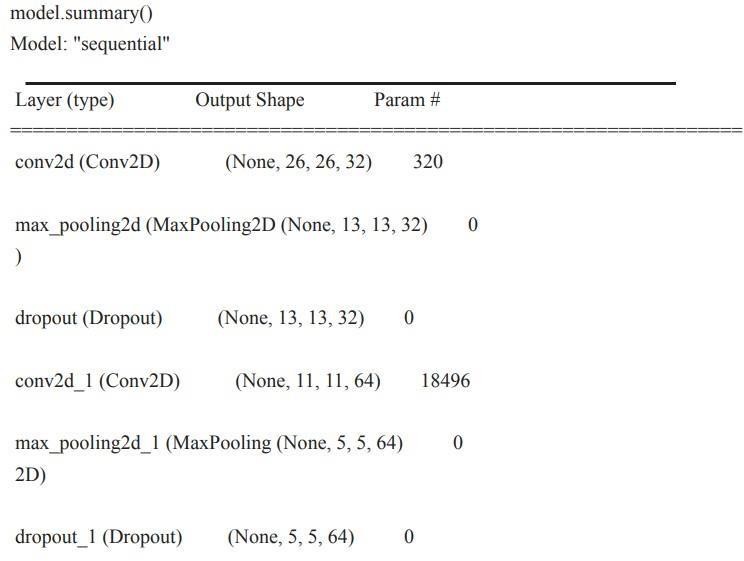
# To classify the images, we still need a Dense and Softmax layer. # We need to flatten the 3x3x128 feature map to a vector of size 1152 keras.layers.Flatten(), keras.layers.Dense(128, activation='relu'),

# 128 Size of Node in Dense Layer

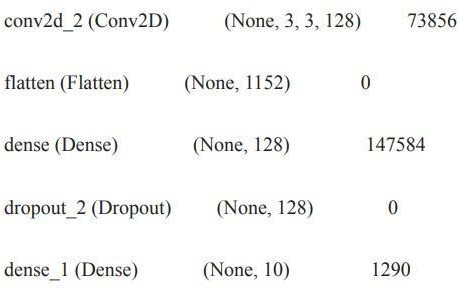
# 1152\*128 = 147584 keras.layers.Dropout(0.25), keras.layers.Dense(10, activation='softmax')

# 10 Size of Node another Dense Layer # 128\*10+10 bias= 1290

])







Total params: 241,546

Trainable params: 241,546

Non-trainable params: 0

# Compile and Train the Model

# After defining the model, we will compile it and train it on the training data. model.compile(optimizer='adam', loss='sparse\_categorical\_crossentropy', metrics=['accuracy']) history = model.fit(x\_train, y\_train, epochs=10, validation\_data=(x\_test, y\_test)) # 1875 is a number of batches. By default batches contain 32 samles.60000 / 32 = 1875 # Finally, we will evaluate the performance of the model on the test data. test\_loss, test\_acc = model.evaluate(x\_test, y\_test)

print('Test accuracy:', test\_acc)

