**Lab Assignment No. 13**

**Code:**

import tensorflow as tf  
from tensorflow import keras  
from keras import Sequential  
from keras.layers import Dense, Dropout, Flatten  
import numpy as np  
import pandas as pd

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

x\_test.shape

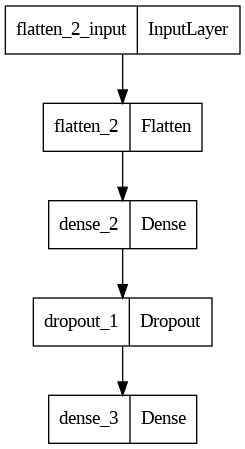
(10000, 28, 28)

x\_test[0][0].shape

(28,)

model = Sequential()  
model.add(Flatten(input\_shape=(28,28)))  
model.add(Dense(128, activation='relu'))  
model.add(Dropout(0.2))  
model.add(Dense(10, activation='softmax'))

from keras.utils import plot\_model  
plot\_model(model)



model.compile(optimizer='adam', loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])

model.fit(x\_train, y\_train, epochs=20)

Epoch 1/20  
1875/1875 [==============================] - 10s 5ms/step - loss: 2.4438 - accuracy: 0.7533  
Epoch 2/20  
1875/1875 [==============================] - 9s 5ms/step - loss: 0.5959 - accuracy: 0.8427  
Epoch 3/20  
1875/1875 [==============================] - 8s 4ms/step - loss: 0.4775 - accuracy: 0.8741  
Epoch 4/20  
1875/1875 [==============================] - 9s 5ms/step - loss: 0.4282 - accuracy: 0.8868  
Epoch 5/20  
1875/1875 [==============================] - 9s 5ms/step - loss: 0.3929 - accuracy: 0.8992  
Epoch 6/20  
1875/1875 [==============================] - 7s 4ms/step - loss: 0.3806 - accuracy: 0.9053  
Epoch 7/20  
1875/1875 [==============================] - 9s 5ms/step - loss: 0.3620 - accuracy: 0.9105  
Epoch 8/20  
1875/1875 [==============================] - 8s 4ms/step - loss: 0.3399 - accuracy: 0.9143  
Epoch 9/20  
1875/1875 [==============================] - 8s 4ms/step - loss: 0.3321 - accuracy: 0.9180  
Epoch 10/20  
1875/1875 [==============================] - 9s 5ms/step - loss: 0.3288 - accuracy: 0.9205  
Epoch 11/20  
1875/1875 [==============================] - 7s 4ms/step - loss: 0.3168 - accuracy: 0.9228  
Epoch 12/20  
1875/1875 [==============================] - 8s 4ms/step - loss: 0.3086 - accuracy: 0.9234  
Epoch 13/20  
1875/1875 [==============================] - 9s 5ms/step - loss: 0.3144 - accuracy: 0.9246  
Epoch 14/20  
1875/1875 [==============================] - 15s 8ms/step - loss: 0.2978 - accuracy: 0.9266  
Epoch 15/20  
1875/1875 [==============================] - 10s 5ms/step - loss: 0.3015 - accuracy: 0.9278  
Epoch 16/20  
1875/1875 [==============================] - 11s 6ms/step - loss: 0.2903 - accuracy: 0.9278  
Epoch 17/20  
1875/1875 [==============================] - 13s 7ms/step - loss: 0.2832 - accuracy: 0.9319  
Epoch 18/20  
1875/1875 [==============================] - 13s 7ms/step - loss: 0.2923 - accuracy: 0.9307  
Epoch 19/20  
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2869 - accuracy: 0.9321  
Epoch 20/20  
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2834 - accuracy: 0.9320

<keras.callbacks.History at 0x7f2103559030>

prediction = model.predict(x\_test)

313/313 [==============================] - 1s 2ms/step

prediction[3].shape

(10,)

x\_test[0][0]

array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0], dtype=uint8)

prediction[0]

array([0.0000000e+00, 1.3933523e-19, 2.9269228e-15, 7.6539842e-12,  
 7.4557314e-31, 4.5651832e-31, 0.0000000e+00, 9.9999994e-01,  
 0.0000000e+00, 5.6528885e-29], dtype=float32)

prediction[0][7]

0.99999994

prediction[1][1]

9.435168e-11

pd.Series(prediction[2]).idxmax()

1

**Output:**

for i in range(10):  
 print(f"Prediction of {prediction[i]} is ", pd.Series(prediction[i]).idxmax())

Prediction of [0.0000000e+00 1.3933523e-19 2.9269228e-15 7.6539842e-12 7.4557314e-31  
 4.5651832e-31 0.0000000e+00 9.9999994e-01 0.0000000e+00 5.6528885e-29] is 7  
Prediction of [0.0000000e+00 9.4351679e-11 9.9999994e-01 1.3839746e-21 0.0000000e+00  
 0.0000000e+00 0.0000000e+00 2.9035965e-19 2.7713451e-30 0.0000000e+00] is 2  
Prediction of [0.0000000e+00 9.9999994e-01 0.0000000e+00 0.0000000e+00 9.8192167e-27  
 8.6102357e-27 1.3392039e-24 6.7198263e-25 3.7141534e-19 0.0000000e+00] is 1  
Prediction of [9.99999940e-01 0.00000000e+00 3.97543002e-18 2.87898494e-15  
 4.17670492e-24 2.52059403e-17 4.16241627e-19 4.58924653e-15  
 8.97595410e-23 1.08165845e-29] is 0  
Prediction of [2.8448460e-23 1.3613716e-11 1.5544450e-14 2.5488617e-13 9.9999994e-01  
 2.5542689e-13 2.4306238e-10 1.3248903e-12 3.4214355e-23 2.7959052e-08] is 4  
Prediction of [0.0000000e+00 9.9999994e-01 0.0000000e+00 0.0000000e+00 2.3557797e-24  
 5.3822781e-25 7.0089274e-24 7.2132002e-25 1.9120865e-19 0.0000000e+00] is 1  
Prediction of [4.9079178e-32 3.3550220e-12 1.7724985e-10 8.9461727e-17 9.9999994e-01  
 6.9597727e-16 2.2915382e-23 1.8836836e-13 1.2865342e-32 6.4851630e-10] is 4  
Prediction of [6.5651999e-14 1.0212370e-12 4.1792095e-08 2.0033461e-03 2.7078322e-05  
 9.8896944e-06 2.2923856e-31 2.4588793e-04 1.6237841e-13 9.9771374e-01] is 9  
Prediction of [1.9678354e-04 2.5814313e-06 6.8850612e-05 5.4609153e-04 1.9435544e-09  
 5.3101850e-01 4.3385461e-01 9.2009661e-10 2.9303946e-02 5.0086309e-03] is 5  
Prediction of [8.8292691e-37 9.5649813e-14 4.9173587e-15 2.1960420e-09 8.8311272e-06  
 3.7559491e-12 2.3588077e-30 1.2012749e-05 1.6326638e-16 9.9997908e-01] is 9

loss, accuracy = model.evaluate(x\_test, y\_test)

313/313 [==============================] - 1s 3ms/step - loss: 0.3086 - accuracy: 0.9480

print(f"Loss of model is {loss} and accuracy of model is {accuracy}")

Loss of model is 0.3086441457271576 and accuracy of model is 0.9480000138282776