**Lab Assignment No. 12**

**Code:**

import tensorflow as tf  
from tensorflow import keras  
from keras import Sequential  
from keras.layers import Conv2D, Dense, MaxPooling2D, Flatten, Dropout, Input  
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
from keras.utils import plot\_model  
from keras.datasets import mnist

num\_classes = 10  
input\_shape = (28, 28, 1)  
(x\_train, y\_train), (x\_test, y\_test)= mnist.load\_data()  
x\_train = x\_train.astype("float32") / 255  
x\_test = x\_test.astype("float32") / 255  
x\_train = np.expand\_dims(x\_train, -1)  
x\_test = np.expand\_dims(x\_test, -1)  
print("x\_train shape:", x\_train.shape)  
print(x\_train.shape[0], "train samples")  
print(x\_test.shape[0], "test samples")  
# convert class vectors to binary class matrices  
y\_train = keras.utils.to\_categorical(y\_train, num\_classes)  
y\_test = keras.utils.to\_categorical(y\_test, num\_classes)

Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz  
11490434/11490434 [==============================] - 1s 0us/step  
x\_train shape: (60000, 28, 28, 1)  
60000 train samples  
10000 test samples

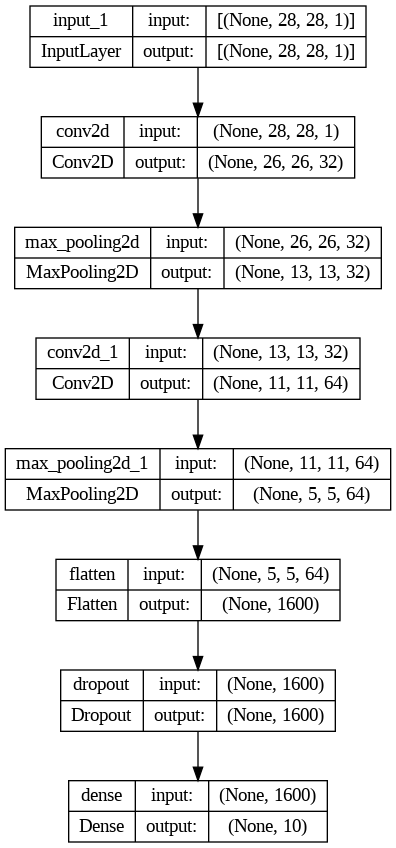
print("len of x\_train: ", len(x\_train), "Len of y\_train: ", len(y\_train), " len of x\_test: ", len(x\_test), " len y\_test: ", len(y\_test))

len of x\_train: 60000 Len of y\_train: 60000 len of x\_test: 10000 len y\_test: 10000

model = keras.Sequential(  
 [  
 keras.Input(shape=input\_shape),  
 Conv2D(32, kernel\_size=(3, 3), activation="relu"),  
 MaxPooling2D(pool\_size=(2, 2)),  
 Conv2D(64, kernel\_size=(3, 3), activation="relu"),  
 MaxPooling2D(pool\_size=(2, 2)),  
 Flatten(),  
 Dropout(0.5),  
 Dense(num\_classes, activation="softmax"),  
 ]  
)  
  
model.summary()  
  
model.compile(optimizer='adam', loss="categorical\_crossentropy", metrics=['accuracy'])

Model: "sequential"  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
 Layer (type) Output Shape Param #   
=================================================================  
 conv2d (Conv2D) (None, 26, 26, 32) 320   
   
 max\_pooling2d (MaxPooling2D (None, 13, 13, 32) 0   
 )   
   
 conv2d\_1 (Conv2D) (None, 11, 11, 64) 18496   
   
 max\_pooling2d\_1 (MaxPooling (None, 5, 5, 64) 0   
 2D)   
   
 flatten (Flatten) (None, 1600) 0   
   
 dropout (Dropout) (None, 1600) 0   
   
 dense (Dense) (None, 10) 16010   
   
=================================================================  
Total params: 34,826  
Trainable params: 34,826  
Non-trainable params: 0  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

plot\_model(model, show\_shapes=True)



x\_train.shape

(60000, 28, 28, 1)

y\_train.shape

(60000, 10)

epochs=10  
batch\_size = 128  
history = model.fit(x\_train, y\_train, batch\_size=batch\_size, epochs=epochs, validation\_split=0.1)

**Output:**

Epoch 1/10  
422/422 [==============================] - 57s 133ms/step - loss: 0.3786 - accuracy: 0.8882 - val\_loss: 0.0825 - val\_accuracy: 0.9783  
Epoch 2/10  
422/422 [==============================] - 49s 117ms/step - loss: 0.1133 - accuracy: 0.9661 - val\_loss: 0.0569 - val\_accuracy: 0.9850  
Epoch 3/10  
422/422 [==============================] - 50s 117ms/step - loss: 0.0847 - accuracy: 0.9744 - val\_loss: 0.0498 - val\_accuracy: 0.9865  
Epoch 4/10  
422/422 [==============================] - 48s 114ms/step - loss: 0.0713 - accuracy: 0.9777 - val\_loss: 0.0415 - val\_accuracy: 0.9890  
Epoch 5/10  
422/422 [==============================] - 47s 111ms/step - loss: 0.0638 - accuracy: 0.9797 - val\_loss: 0.0436 - val\_accuracy: 0.9885  
Epoch 6/10  
422/422 [==============================] - 48s 114ms/step - loss: 0.0574 - accuracy: 0.9824 - val\_loss: 0.0360 - val\_accuracy: 0.9910  
Epoch 7/10  
422/422 [==============================] - 51s 120ms/step - loss: 0.0512 - accuracy: 0.9841 - val\_loss: 0.0384 - val\_accuracy: 0.9883  
Epoch 8/10  
422/422 [==============================] - 48s 114ms/step - loss: 0.0476 - accuracy: 0.9846 - val\_loss: 0.0321 - val\_accuracy: 0.9905  
Epoch 9/10  
422/422 [==============================] - 53s 125ms/step - loss: 0.0439 - accuracy: 0.9861 - val\_loss: 0.0302 - val\_accuracy: 0.9918  
Epoch 10/10  
422/422 [==============================] - 50s 117ms/step - loss: 0.0434 - accuracy: 0.9862 - val\_loss: 0.0315 - val\_accuracy: 0.9908

prediction = model.predict(x\_test)

313/313 [==============================] - 3s 9ms/step

prediction.shape

(10000, 10)

y\_test.shape

(10000, 10)

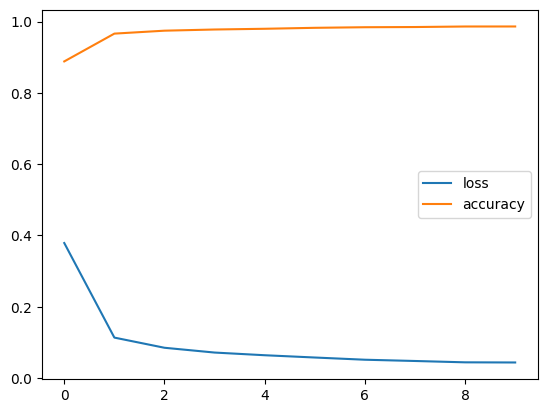
loss, accuracy = model.evaluate(x\_test, y\_test)

313/313 [==============================] - 3s 9ms/step - loss: 0.0276 - accuracy: 0.9908

print(f"Loss of model is on testing data: {loss} and accuracy of model is on testing data: {accuracy}")

Loss of model is on testing data: 0.027647219598293304 and accuracy of model is on testing data: 0.9908000230789185

import matplotlib.pyplot as plt  
  
plt.plot(history.history['loss'], label='loss')  
plt.plot(history.history['accuracy'], label='accuracy')  
plt.legend()  
plt.show()



plt.plot(history.history['val\_loss'], label='Val loss')  
plt.plot(history.history['val\_accuracy'], label='Val accuracy')  
plt.legend()  
plt.show()

