PRACTICAL-2

PROBLEM STATEMENT:

2. Implementing Feedforward neural networks with Keras and TensorFlow

a. Import the necessary packages

b. Load the training and testing data (MNIST/CIFAR10)

c. Define the network architecture using Keras

d. Train the model using SGD

e. Evaluate the network

f. Plot the training loss and accuracy

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CODE\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# a. IMPORTING NECESSARY PACKAGES

import tensorflow as tf

from tensorflow import keras

import matplotlib.pyplot as plt

import random

# b. LOAD THE TRAINING AND TESTING DATA (MNIST)

mnist = tf.keras.datasets.mnist

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

# Normalize the images

x\_train = x\_train / 255.0

x\_test = x\_test / 255.0

# Check the shape of the input images

img\_len, img\_width = x\_train.shape[1:3]

print('Size of input image:', img\_len, 'x', img\_width)

# c. DEFINE THE NETWORK ARCHITECTURE USING KERAS

model = keras.Sequential([

    keras.layers.Flatten(input\_shape=(28, 28)),

    keras.layers.Dense(128, activation="relu"),

    keras.layers.Dense(10, activation="softmax")

])

# Display the model architecture

model.summary()

# d. TRAIN THE MODEL USING SGD

model.compile(optimizer="sgd",

              loss="sparse\_categorical\_crossentropy",

              metrics=['accuracy'])

history = model.fit(x\_train, y\_train, validation\_data=(x\_test, y\_test), epochs=3)

# e. EVALUATE THE NETWORK

test\_loss, test\_acc = model.evaluate(x\_test, y\_test)

print("Loss=%.3f" % test\_loss)

print("Accuracy=%.3f" % test\_acc)

# Display a random test image

n = random.randint(0, len(x\_test) - 1)

plt.imshow(x\_test[n], cmap='gray')

plt.title(f"Label: {y\_test[n]}")

plt.show()

# f. PLOT THE TRAINING LOSS AND ACCURACY

# Plotting the Training and Validation Accuracy

plt.plot(history.history['accuracy'], label='Train Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.title('Model Accuracy')

plt.xlabel('Epoch')

plt.ylabel('Accuracy')

plt.legend(loc='upper right')

plt.show()

# Plotting the Training and Validation Loss

plt.plot(history.history['loss'], label='Train Loss')

plt.plot(history.history['val\_loss'], label='Validation Loss')

plt.title('Model Loss')

plt.xlabel('Epoch')

plt.ylabel('Loss')

plt.legend(loc='upper left')

plt.show()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*