# ------------------------ Step 1: Import Necessary Libraries ------------------------

import tensorflow as tf # TensorFlow is the main deep learning framework

from tensorflow.keras.datasets import mnist # MNIST dataset (handwritten digits) is loaded from Keras

from tensorflow.keras.models import Sequential # Sequential model is used to stack layers

from tensorflow.keras.layers import Dense, Flatten # Dense = fully connected layer, Flatten = convert 2D image to 1D

from tensorflow.keras.optimizers import SGD # SGD = Stochastic Gradient Descent optimizer

import matplotlib.pyplot as plt # For plotting graphs of accuracy and loss

# ------------------------ Step 2: Load the Dataset ------------------------

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data() # Load training & testing data

# ------------------------ Step 3: Preprocessing the Data ------------------------

# Normalize pixel values from range [0,255] to [0,1] for faster learning

x\_train = x\_train / 255.0

x\_test = x\_test / 255.0

# ------------------------ Step 4: Define Neural Network Model ------------------------

model = Sequential([ # Sequential model = layers are added one by one

Flatten(input\_shape=(28, 28)), # Converts 28x28 image into 1D vector of 784 values

Dense(128, activation='relu'), # Hidden layer 1 with 128 neurons using ReLU activation

Dense(64, activation='relu'), # Hidden layer 2 with 64 neurons

Dense(10, activation='softmax') # Output layer with 10 classes (digits 0-9), softmax gives probabilities

])

# ------------------------ Step 5: Compile the Model ------------------------

model.compile(

optimizer=SGD(learning\_rate=0.01), # Optimizer = Stochastic Gradient Descent with learning rate 0.01

loss='sparse\_categorical\_crossentropy', # Loss function for multi-class classification

metrics=['accuracy'] # We want to track accuracy during training/testing

)

# ------------------------ Step 6: Train the Model ------------------------

history = model.fit(

x\_train, y\_train, # Training data

epochs=10, # How many times the full dataset will pass through the model

batch\_size=32, # Number of samples processed before the model is updated

validation\_data=(x\_test, y\_test) # Use test data to validate after every epoch

)

# ------------------------ Step 7: Evaluate the Model ------------------------

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

print("Test Loss:", test\_loss)

print("Test Accuracy:", test\_accuracy)

# ------------------------ Step 8: Plot Training Loss and Accuracy ------------------------

# Accuracy Plot

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

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

plt.title("Model Accuracy")

plt.xlabel("Epochs")

plt.ylabel("Accuracy")

plt.legend()

plt.show()

# Loss Plot

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

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

plt.title("Model Loss")

plt.xlabel("Epochs")

plt.ylabel("Loss")

plt.legend()

plt.show()

**2. Explanation of Each Step in Simple Words**

| **Step** | **Explanation** |
| --- | --- |
| Import Libraries | We import TensorFlow, MNIST dataset, layers like Dense and Flatten, optimizer (SGD), and plotting library. |
| Load Data | mnist.load\_data() gives training & test images of handwritten digits (0–9). |
| Preprocess | Images are 28×28 pixels. Pixel values are divided by 255 to convert from [0–255] to [0–1]. |
| Model Architecture | Flatten → Dense(128 ReLU) → Dense(64 ReLU) → Dense(10 Softmax). |
| Compile | Model needs an optimizer (SGD), a loss function, and metric (accuracy). |
| Training | Model is trained for 10 epochs using mini-batches of 32 samples. |
| Evaluate | After training, test data is used to check performance. |
| Plot | Graphs of accuracy and loss across 10 epochs are shown. |