#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.

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import layers

import numpy as np

import random

import matplotlib.pyplot as plt

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

# Load and preprocess MNIST dataset

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

# Normalize pixel values to be between 0 and 1

x\_train, x\_test = x\_train / 255.0, x\_test / 255.0

#c. Define the network architecture using Keras.

model = keras.Sequential([

    layers.Flatten(input\_shape=(28, 28)),  # Flatten the 28x28 input images

    layers.Dense(128, activation='relu'),   # Hidden layer with 128 units and ReLU activation

    layers.Dropout(0.2),                   # Dropout layer to prevent overfitting

    layers.Dense(10, activation='softmax') # Output layer with 10 units for 10 classes and softmax activation

])

model.summary()

plt.matshow(x\_train[0])

#d. Train the model using SGD.

# Compile the model

model.compile(optimizer='sgd',  # Stochastic Gradient Descent

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

              metrics=['accuracy'])

# Train the model

history = model.fit(x\_train, y\_train, epochs=5, validation\_data=(x\_test, y\_test))

#e. Evaluate the network.

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

print(f'Test accuracy: {test\_accuracy}')

#f. Plot the training loss and accuracy.

# Plot training history

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)

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

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

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

plt.subplot(1, 2, 2)

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

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

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.legend()

plt.show()