First, I will import the necessary libraries and load the MNIST dataset using the load\_data() function from the tf.keras.datasets module. I will then split the dataset into training and test sets and preprocess the data by reshaping it into a 2D array and scaling it between 0 and 1.

# Import necessary libraries import numpy as np import tensorflow as tf from tensorflow.keras import Sequential from tensorflow.keras.layers import Dense from tensorflow.keras.optimizers import SGD # Load MNIST dataset (X\_train, y\_train), (X\_test, y\_test) = tf.keras.datasets.mnist.load\_data() # Preprocess data X\_train = X\_train.reshape(X\_train.shape[0], -1) / 255.0 X\_test = X\_test.reshape(X\_test.shape[0], -1) / 255.0 # Define MLP models with different activation functions model\_sigmoid = Sequential() model\_sigmoid.add(Dense(units=64, activation='sigmoid', input\_shape= (X\_train.shape[1],))) model\_sigmoid.add(Dense(units=10, activation='sigmoid')) model\_relu = Sequential() model\_relu.add(Dense(units=64, activation='relu', input\_shape=(X\_train.shape[1],))) model\_relu.add(Dense(units=10, activation='relu')) model\_tanh = Sequential() model\_tanh.add(Dense(units=64, activation='tanh', input\_shape=(X\_train.shape[1],))) model\_tanh.add(Dense(units=10, activation='tanh')) # Compile models model\_sigmoid.compile(optimizer=SGD(lr=0.1), loss='sparse\_categorical\_crossentropy', metrics=['accuracy']) model\_relu.compile(optimizer=SGD(lr=0.1), loss='sparse\_categorical\_crossentropy', metrics=['accuracy']) model\_tanh.compile(optimizer=SGD(lr=0.1), loss='sparse\_categorical\_crossentropy', metrics=['accuracy']) # Train models history\_sigmoid = model\_sigmoid.fit(X\_train, y\_train, epochs=5, batch\_size=32, validation\_data=(X\_test, y\_test)) history\_relu = model\_relu.fit(X\_train, y\_train, epochs=5, batch\_size=32, validation\_data=(X\_test, y\_test)) history\_tanh = model\_tanh.fit(X\_train, y\_train, epochs=5, batch\_size=32, validation\_data=(X\_test, y\_test)) # Evaluate models sigmoid\_loss, sigmoid\_acc = model\_sigmoid.evaluate(X\_test, y\_test) relu\_loss, relu\_acc = model\_relu.evaluate(X\_test, y\_test) tanh\_loss, tanh\_acc = model\_tanh.evaluate(X\_test, y\_test) # Print test set evaluation metrics print(f'Sigmoid model: loss = {sigmoid\_loss:.4f}, accuracy = {sigmoid\_acc:.4f}') print(f'ReLU model: loss = {relu\_loss:.4f}, accuracy = {relu\_acc:.4f}') print(f'Tanh model: loss = {tanh\_loss:.4f}, accuracy = {tanh\_acc:.4f}') # Plot loss and accuracy curves import matplotlib.pyplot as plt plt.figure(figsize=(12, 6)) plt.subplot(1, 2, 1) plt.plot(history\_sigmoid.history['loss'], label='Sigmoid') plt.plot(history\_relu.history['loss'], label='ReLU') plt.plot(history\_tanh.history['loss'], label='Tanh') plt.title('Loss curve') plt.xlabel('Epoch') plt.ylabel('Loss') plt.legend() plt.subplot(1, 2, 2) plt.plot(history\_sigmoid.history['accuracy'], label='Sigmoid') plt.plot(history\_relu.history['accuracy'], label='ReLU') plt.plot(history\_tanh.history['accuracy'], label='Tanh') plt.title('Accuracy curve') plt.xlabel('Epoch') plt.ylabel('Accuracy') plt.legend() plt.show()