NANDHA ENGINEERING COLLEGE ERODE - 638052

(An Autonomous Institution, Affiliated to Anna University, Chennai)



# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**22CSX01 – Deep Learning Assignment - ISubmitted by**

**TEAM - 6**

# PRIYADARSHIKAA E (22CS069)

**RAGAVENDIRAN S (22CS070)**

1.You are evaluating whether to use ReLU or ELU as the activation

function in a CNN for classifying images of handwritten digits. How would you compare their performance?

# Signature of the students Signature of the faculty

22CS069 :

22CS070 :

**You are evaluating whether to use ReLU or ELU as the activation function in a CNN for classifying images of handwritten digits. How would you compare their performance?**

**Program:**

# Import necessary libraries import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense from tensorflow.keras.datasets import mnist

import matplotlib.pyplot as plt

# Load and preprocess the dataset

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data() x\_train, x\_test = x\_train / 255.0, x\_test / 255.0 x\_train = x\_train.reshape(-1, 28, 28, 1)

x\_test = x\_test.reshape(-1, 28, 28, 1)

# Convert labels to categorical one-hot encoding y\_train = tf.keras.utils.to\_categorical(y\_train, 10) y\_test = tf.keras.utils.to\_categorical(y\_test, 10)

# Function to create a CNN model with a specified activation function def create\_model(activation\_function):

model = Sequential([

Conv2D(32, (3, 3), activation=activation\_function, input\_shape=(28, 28, 1)),

MaxPooling2D((2, 2)),

Conv2D(64, (3, 3), activation=activation\_function),

MaxPooling2D((2, 2)), Flatten(),

Dense(128, activation=activation\_function), Dense(10, activation='softmax')

])

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy']) return model

# Create models for ReLU and ELU relu\_model = create\_model('relu') elu\_model = create\_model('elu')

# Train and evaluate ReLU model print("Training ReLU model...")

history\_relu = relu\_model.fit(x\_train, y\_train, epochs=10, batch\_size=128, validation\_data=(x\_test, y\_test))

# Train and evaluate ELU model print("Training ELU model...")

history\_elu = elu\_model.fit(x\_train, y\_train, epochs=10, batch\_size=128, validation\_data=(x\_test, y\_test))

# Plot training and validation accuracy plt.figure(figsize=(12, 5))

# Accuracy comparison plt.subplot(1, 2, 1)

plt.plot(history\_relu.history['accuracy'], label='ReLU Training Accuracy') plt.plot(history\_relu.history['val\_accuracy'], label='ReLU Validation Accuracy') plt.plot(history\_elu.history['accuracy'], label='ELU Training Accuracy')

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

plt.ylabel('Accuracy') plt.legend()

plt.title('Training and Validation Accuracy')

# Loss comparison plt.subplot(1, 2, 2)

plt.plot(history\_relu.history['loss'], label='ReLU Training Loss') plt.plot(history\_relu.history['val\_loss'], label='ReLU Validation Loss') plt.plot(history\_elu.history['loss'], label='ELU Training Loss') plt.plot(history\_elu.history['val\_loss'], label='ELU Validation Loss') plt.xlabel('Epoch')

plt.ylabel('Loss') plt.legend()

plt.title('Training and Validation Loss') plt.show()

# Print final evaluation results

print("ReLU Model Test Loss & Accuracy:", relu\_model.evaluate(x\_test, y\_test)) print("ELU Model Test Loss & Accuracy:", elu\_model.evaluate(x\_test, y\_test))

**Output:**

