Machine Learning
LABORATORY: Deep Neural Network

NAME:	STUDENT ID#:

Objectives:

- Understand and implement the forward pass of a neural network using matrix operations.
- Apply activation functions based on textbook equations to non-linearize the model.
- Use softmax and cross-entropy loss for multi-class classification tasks.
- Experiment with different layer configurations to observe the effect on model accuracy.
- Evaluate classifier performance using confusion matrix, ROC, and standard metrics (precision, recall, accuracy, F1-score).

Part 1. Instruction

- In this assignment, you will build a multilayer feedforward neural network using **only NumPy and Matplotlib** to solve a **multi-class classification** problem on the **MNIST dataset**. You **must** use the **dataset provided by the TA.** Use only **NumPy** for matrix operations and **Matplotlib/Seaborn** for plotting. Implement the forward pass only, without backpropagation.
- Follow the arithmetic instructions and code template provided in Part 3 of this lab. Evaluate your model using accuracy, confusion matrix, and ROC curves.
- You need to try experiment with different activation functions and network structures to **improve your model's performance.**

Part 2. Arithmetic Instructions.

Step **Procedure**

1 Activation Function- Refer to equation 6.14 – 6.18

2 Multilayer Feedforward
- Refer to Equation 6.7-6.9
- Equation 6.19:

$$z^{(l)} = h^{(l)}(W^{(l)} z^{(l-1)})$$

3 Cross Entropy and one-hot encoding – Equation 6.36:

$$E(\mathbf{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{kn} \ln\, y_k(\mathbf{x}_n, \mathbf{w})$$

SoftMax Function – Equation 6.37:

$$y_k(\mathbf{x}, \mathbf{w}) = \frac{\exp\,(a_k(\mathbf{x}, \mathbf{w}))}{\sum_j \exp\,\left(a_j(\mathbf{x}, \mathbf{w})\right)}$$

5 Evaluation using Confusion Matrix, ROC, Accuracy, Precision, Recall
Refer to the previous Labs

Lecture: Prof. Hsien-I Lin
TA: Satrio Sanjaya and Muhammad Ahsan

Part 3. Data Transfer Instructions.

Step	Procedure

1 #Load Dataset

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import struct
import pandas as pd

# === Step 1: Load MNIST Dataset ===
def load_mnist_images(filename):
    with open(filename, 'rb') as f:
        _, num, rows, cols = struct.unpack(">IIII", f.read(16))
        images = np.frombuffer(f.read(), dtype=np.uint8).reshape(num, rows * cols)
        return images / 255.0

def load_mnist_labels(filename):
    with open(filename, 'rb') as f:
        _, num = struct.unpack(">II", f.read(8))
        labels = np.frombuffer(f.read(), dtype=np.uint8)
        return labels
 # Students can experiment to modify number of Train
X_train = load_mnist_images("train-images.idx3-ubyte__")[:500]
y_train = load_mnist_labels("train-labels.idx1-ubyte__")[:500]
X_test = load_mnist_images("t10k-images.idx3-ubyte__")[:200]
y_test = load_mnist_labels("t10k-labels.idx1-ubyte__")[:200]
```

2

```
# === Step 2: Activation Functions (Refer to Eq. 6.14 - 6.18) ===
def relu(x): return None
def tanh(x): return None
def softplus(x): return None
def leaky_relu(x, alpha=0.1): return None

def one_hot(y, num_classes=10): # Refer to Equation 6.36
    return None
def cross_entropy(y_pred, y_true): # Refer to Equation 6.36
    return None
def softmax(a): # Refer to Equation 6.37
    return None

def forward_pass(X, weights, activations): # Forward Pass (Eq. 6.19) ===
    return None
```

3

```
# === Step 3: Training Loop === # Students can experiment to modify
np.random.seed(42)
input_size = 784
hidden1 = 64
hidden2 = 32
output_size = 10
epochs = 30
best_loss = float('inf')
best_weights = None
```

```
for epoch in range(epochs):
    # TODO: Randomly initialize weights for each layer
    W1 = None
    W2 = None
    W3 = None

    weights = [W1, W2, W3]
    activations = [relu, relu, softmax]  # Students can experiment to modify
```

4

```
# === Step 4: Evaluation Metrics (Confusion Matrix, ROC, etc) ===
def compute_confusion_matrix(y_true, y_pred, num_classes=10): return None

# === ROC Curve ===
def compute_roc(y_true_oh, y_pred_proba):  return None

# === Classification Report === Print TP, FP, FN, TN, precision, recall, f1, accuracy
def compute_metrics(cm): return None

print("=== Classification Report === Print TP, FP, FN, TN, precision, recall, f1 for each
class and overall accuracy")
```

Grading Assignment & Submission (70% Max)

Implementation (50%):

1. (10%) Implement a Feedforward Neural Network with More Than One Hidden Layer.
2. (10%) Activation functions implemented from scratch (Eq. 6.14–6.18), and Softmax output and cross-entropy loss (Eq. 6.36 & 6.37).
3. (15%) Model runs correctly and generates prediction results.
4. Evaluation:
    a. (5%) Confusion matrix (plotted),
    b. (5%) ROC curve for 10 classes (plotted),
    c. (5%) Precision, Recall, F1, Overall Accuracy

Question (20%):

1. Explain how you designed your model (number of layers, neurons, and activation functions). What changes did you make to improve the accuracy, and how did those changes affect the results? *Please attach the performance results before and after your improvements.
2. Based on your evaluation results (confusion matrix, ROC, etc.), how well did the model perform? Which classes are harder to predict? Why do you think that happened?

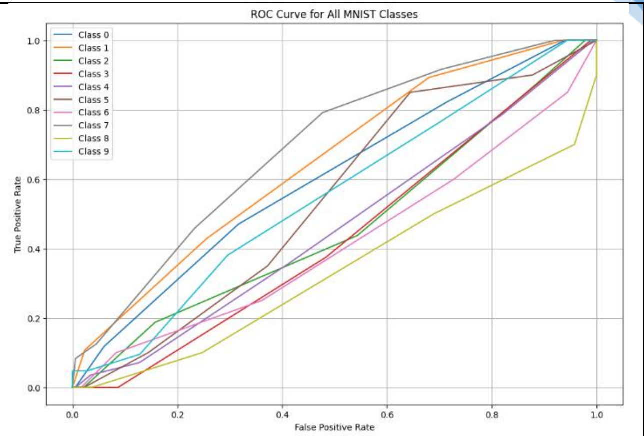
Submission :

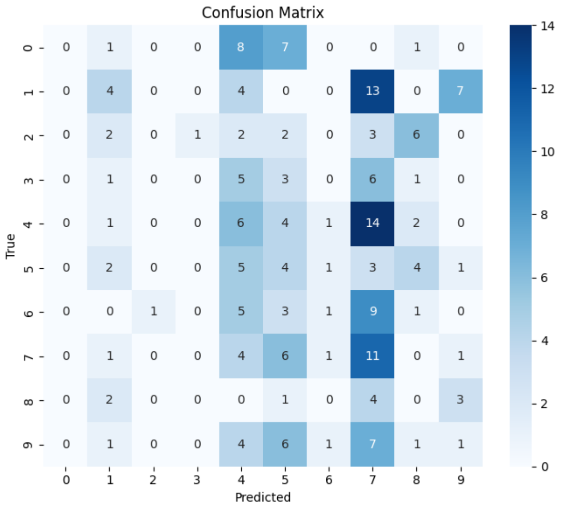
1. Report: Answer all conceptual questions. Include screenshots of your results in the last pages of this PDF File.
2. Code: Submit your complete Python script in either .py or .ipynb format.
3. Upload both your report and code to the E3 system (**Labs3 Homework Assignment**). Name your files correctly:
    a. Report: StudentID_Lab3_Homework.pdf
    b. Code: StudentID_Lab3_ Homework.py or StudentID_Lab3_ Homework.ipynb
4. Deadline: Sunday – 21:00 PM
5. Plagiarism is **strictly prohibited**. Submitting copied work from other students will result in penalties.

Lecture: Prof. Hsien-I Lin
TA: Satrio Sanjaya and Muhammad Ahsan

Example Output (Just for reference):

```
Class 1:
  TP: 4, FP: 11, FN: 24, TN: 161
  Precision: 0.2667
  Recall:    0.1429
  F1 Score:  0.1860

Class 2:
  TP: 0, FP: 1, FN: 16, TN: 183
  Precision: 0.0000
  Recall:    0.0000
  F1 Score:  0.0000

Class 3:
  TP: 0, FP: 1, FN: 16, TN: 183
  Precision: 0.0000
  Recall:    0.0000
  F1 Score:  0.0000

Class 4:
  TP: 6, FP: 37, FN: 22, TN: 135
  Precision: 0.1395
  Recall:    0.2143
  F1 Score:  0.1690

Class 5:
  TP: 4, FP: 32, FN: 16, TN: 148
  Precision: 0.1111
  Recall:    0.2000
  F1 Score:  0.1429

Class 6:
  TP: 1, FP: 4, FN: 19, TN: 176
  Precision: 0.2000
  Recall:    0.0500
  F1 Score:  0.0800

Class 7:
  TP: 11, FP: 59, FN: 13, TN: 117
  Precision: 0.1571
  Recall:    0.4583
  F1 Score:  0.2340

Class 8:
  TP: 0, FP: 16, FN: 10, TN: 174
  Precision: 0.0000
  Recall:    0.0000
  F1 Score:  0.0000

Class 9:
  TP: 1, FP: 12, FN: 20, TN: 167
  Precision: 0.0769
  Recall:    0.0476
  F1 Score:  0.0588

Overall Accuracy: 0.1350
```





Code Results and Answer: