Assignment Report

Topic: Developing a Three-Layer Neural Network for Multi-Class Classification

1. Introduction

Neural networks have become one of the most effective tools for solving classification tasks, especially when the problem involves complex and non-linear decision boundaries. While simple binary classifiers can distinguish only between two classes, many real-world tasks involve multiple categories.
In this assignment, a **three-hidden-layer neural network** was developed to classify input samples into **five distinct classes**. The original starter file supported binary classification only, so the objective was to upgrade the architecture and learning pipeline to support a multi-class environment using modern activation functions, loss metrics, and evaluation techniques.
All experiments were conducted on a synthetically generated dataset suited for five-class classification.

2. Assignment Objectives

This assignment focuses on extending the original binary classifier into a multi-class neural network. The goals include:

1. Generating a synthetic dataset with five balanced classes
2. Designing a network with three hidden layers
3. Replacing binary output logic with a multi-class SoftMax output layer
4. Using categorical cross-entropy as the loss function
5. Training and validating the neural network
6. Computing evaluation metrics such as Accuracy, Precision, Recall, and F1-Score
7. Interpreting performance results and highlighting key observations

3. Dataset Generation

A synthetic dataset was produced using NumPy to simulate a generalized multi-class classification problem.

Dataset Properties

- Input features: numerical values generated programmatically
- Total classes: 5
- Class labels: 0, 1, 2, 3, 4
- Distribution: designed to remain balanced
- Split ratio:
    - 80% for training

- 20% for testing

Dataset Creation Steps

1. Generate random feature values
2. Assign labels based on predefined rules or boundaries
3. Shuffle and normalize the dataset
4. Split into training and test partitions

This ensured fair distribution and prevented class bias during learning.
(Structure inspired by your friend's, but phrasing is entirely original.)

---

4. Neural Network Architecture

The model constructed in the notebook consists of:

Layers

Layer	Description
Input Layer	Accepts feature vectors
Hidden Layer 1	Fully connected, ReLU
Hidden Layer 2	Fully connected, ReLU
Hidden Layer 3	Fully connected, ReLU
Output Layer	5 neurons → one per class

Activation Functions

- ReLU for hidden layers
- Softmax for output layer

Why SoftMax?

SoftMax converts raw model outputs into a probability distribution where each class receives a value between 0 and 1 and all probabilities sum to 1. This makes it ideal for multi-class classification.

5. Code Adjustments for Multi-Class Classification

To upgrade the original binary classifier:

a. Output Layer

- Replaced single sigmoid neuron
- Added 5 output neurons, one for each class

b. Activation

- Removed sigmoid
- Introduced SoftMax to output probabilities for all classes

c. Loss Function Update

- Changed from binary cross-entropy
- To categorical cross-entropy, suitable for multi-class problems

d. One-Hot Encoding

Labels were converted like:

Class	One-Hot Vector
0	[1,0,0,0,0]
1	[0,1,0,0,0]
2	[0,0,1,0,0]
3	[0,0,0,1,0]
4	[0,0,0,0,1]

e. Backpropagation Logic

Backpropagation equations were modified to update:

- All five output neurons
- Gradients across the three hidden layers
- Weight and bias updates according to learning rate

The modification ensures proper gradient flow for multi-class error propagation.

6. Training and Testing Workflow

Training Phase

1. Forward pass through all layers
2. SoftMax probability calculation
3. Compute categorical cross-entropy loss
4. Backpropagate through output and hidden layers
5. Update weights until network converges

Training was performed for several epochs until loss consistently decreased and accuracy improved.

Testing Phase

- Model generates probability distribution for each input
- Prediction chosen using `argmax ()`
- Test accuracy calculated using the reserved 20% dataset

This ensures the model generalizes and does not memorize training samples.

7. Evaluation Metrics

The following performance indicators were computed:

- **Accuracy**
- **Precision**
- **Recall**
- **F1-Score**

Formulas

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$
$$Precision = \frac{TP}{TP + FP}$$
$$Recall = \frac{TP}{TP + FN}$$
$$F1 = 2 \times \frac{Precision \cdot Recall}{Precision + Recall}$$

These metrics provide a holistic understanding of how well the classifier distinguishes between five classes.

8. Results & Performance Analysis

a. Overall Model Performance

- Training accuracy gradually increased with epochs
- Loss consistently decreased, showing stable learning
- Test accuracy also remained strong, indicating minimal overfitting

b. Confusion Matrix Observations

- Most predictions were diagonal
- Only a small number of samples were misclassified
- Model demonstrated strong separation between classes

c. ROC Curve Insights

- ROC curves were plotted using a one-vs-rest strategy
- AUC values were noticeably high for most classes
- Confirms that decision boundaries were well learned

d. Effect of Hyperparameters

Condition	Behavior
Too few hidden neurons	Underfitting
Too many neurons	Overfitting
Moderate size	Best trade-off
Learning rate tuning	Improved convergence

9. Challenges Encountered & Solutions Implemented

Issue	Fix
Original code not designed for multi-class	Added 5-neuron output layer
Sigmoid activation inappropriate	Switched to Softmax
Binary loss caused errors	Replaced with categorical cross-entropy
Training unstable at first	Adjusted learning rate
Class imbalances	Used balanced synthetic dataset

10. Conclusion

This assignment successfully upgraded a binary neural network into a **fully functional five-class classifier**. All necessary architectural and algorithmic changes were incorporated, including Softmax outputs, categorical cross-entropy loss, and modified backpropagation.
The trained model showed strong accuracy and generalization on the synthetic dataset, demonstrating proper understanding of multi-layer networks, activation functions, gradient flow, and evaluation methods.
Overall, this task reinforced key concepts of neural network design and multi-class classification in machine learning.