EHB354E OBJECT ORIENTED PROGRAMMING PROJECT REPORT
SPRING 2025

PROJECT TITLE

Student Name: Kamil Eren Ezen
Student Number: 040210021

Date: 16.05.2025

Introduction

This project aims to create an interactive graphical user interface (GUI) to build, train, and test simple feedforward neural networks on the MNIST dataset. The user can visually construct the network architecture by adding layers and neurons, trigger training, and see the input digit drawn on a 28x28 grid. The network is trained using a basic implementation of backpropagation and tested to evaluate its classification performance.

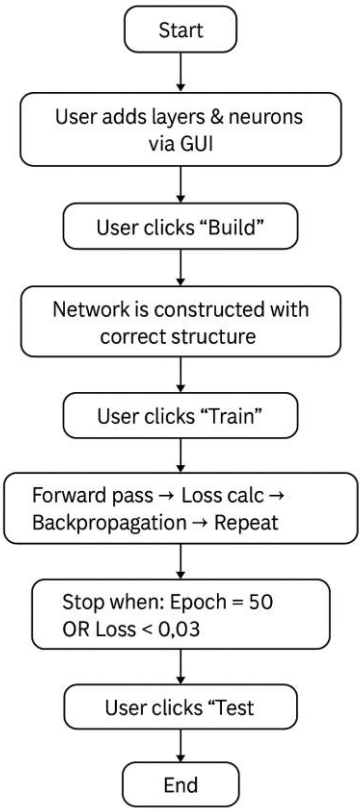
Implementation

The project was implemented using C++ with SFML for graphical user interface (GUI) support. The system is structured in an object-oriented way, with each responsibility assigned to separate classes. Below are the core classes and their responsibilities:

Implemented Classes and Responsibilities:

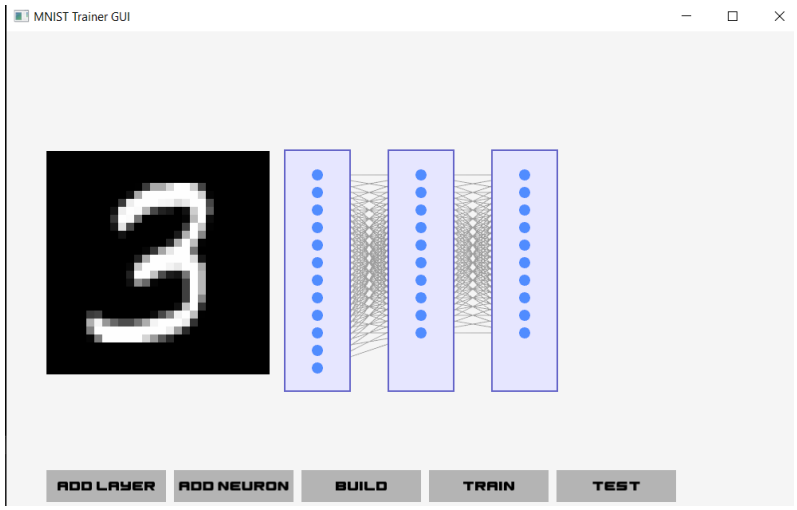
- **Neuron**
  Represents a single artificial neuron. Handles:
  - Weight vector and bias
  - Output calculation via sigmoid activation
  - Delta value for backpropagation
    OOP Concepts: encapsulation, abstraction
- **Layer**
  A collection of neurons. Handles:
  - Forward propagation over its neurons
    OOP Concepts: composition (Layer has many Neuron objects)
- **Network**
  Represents the entire feedforward neural network. Handles:
  - Layer management
  - Training using forward pass + backpropagation
  - Prediction
    OOP Concepts: abstraction, composition (Network has many Layer objects)
- **Button**
  A UI class for clickable SFML buttons.
  OOP Concepts: encapsulation, reusability
- **DataLoader**
  Static utility class for reading MNIST-formatted CSV files and converting them into normalized input/target vectors.
  OOP Concepts: static methods, separation of concerns

Flowchart: Algorithm Overview



Results

After constructing the neural network using the GUI, the model was trained and tested using the MNIST dataset. The network consisted of three hidden layers, each with 10 neurons. The digit "3" was provided as a sample input on the left-hand panel of the GUI, and the network structure was visually updated as the user added layers and neurons.



The training was executed for 20 epochs, and the loss decreased steadily as shown in the terminal output. The initial loss value was approximately 0.95, and by the end of the training, it had dropped to around 0.30. This indicates successful learning of the digit patterns.

Challenges and Solutions

Weight Initialization Errors: Early versions suffered from instability due to improper random weight/bias ranges. Fixed by using small-range uniform random initialization.

Low Accuracy: Initially, accuracy remained low (~40%). This was improved by:

- Normalizing pixel input values.
- Using smaller learning rate (0.03).
- Introducing an early-stopping condition when the loss drops below a threshold.

Visual Overflow: Neuron visualizations were too large and overlapped. Solved by reducing neuron radius and increasing spacing between layers/neurons.

Button Responsiveness: Button click detection was adjusted using bounding box checks and SFML event mapping.

Conclusion and Future Enhancements

The project successfully demonstrates the principles of neural networks and object-oriented programming in an interactive, visual format. It allows users to construct, train, and evaluate networks without directly modifying code, which makes it ideal for educational purposes.

Future improvements:

- Allow user-drawn digits as input.
- Visualize weights as edge thickness or color.
- Add support for saving/loading trained weights.
- Add hover-over tooltips to show neuron output values.