Neural Network Simulation - Project Description

1. Project Goal

The objective of this project is to implement a simple feedforward neural network using C++. The implementation will focus on using object-oriented design principles, operator overloading, STL containers, and efficient matrix computations. The neural network will support forward propagation, backpropagation, and gradient descent for training. The implementation will also include Google Test for unit testing.

2. Class Structure

- Matrix: Represents matrix operations (addition, multiplication, etc.).
- Layer: Represents a neural network layer, holding weights and biases.
- NeuralNetwork: Manages the training and inference of the network.
- ActivationFunction (abstract class): Defines activation function interface.
  - SigmoidFunction: Implements the sigmoid activation function.
  - ReLUFunction: Implements the ReLU activation function.
- Trainer: Handles training data and updates weights.

3. Implementation Plan

1. Convert the existing C Matrix structure into a C++ class.
2. Implement the Layer and NeuralNetwork classes with STL containers.
3. Implement forward propagation using matrix multiplication and activation functions.
4. Implement backpropagation and gradient descent for training.
5. Use STL algorithms (e.g., transform(), for_each()) for efficient operations.
6. Implement file handling to save/load trained models.
7. Implement random initialization of weights using std::random.
8. Write unit tests using Google Test for Matrix and NeuralNetwork functionalities.

4. Testing Approach with Google Test

The Google Test framework will be used for unit testing. Test cases will be written for:

- Matrix class (addition, multiplication, transpose, etc.).
- Activation functions (Sigmoid, ReLU).
- NeuralNetwork (forward propagation, backpropagation, weight updates).

A CMake-based test setup will be provided to automate test execution.