# Neural Network (Project Code: N)

Tajkia Nuri Ananna (001661229), Nowshin Tasnim (001649184),
Ali Salehi Darjani (001657808), Joy Saha (001657938)

March 12, 2025

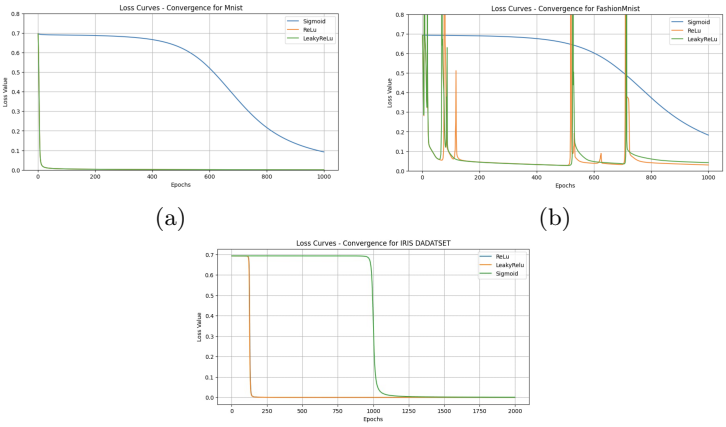
## 1 Problem Statement

Pre-built neural network libraries hide the details of model training and prediction, making it difficult for learners to understand the core principles of neural networks. This project bridges that gap by implementing a neural network from scratch in Python for binary classification, using the datasets MNIST, FashionMNIST, Iris, Dexter,Madelon, Spambase, and Leukemia. Our goal is to explore the impact of different activation functions (Sigmoid, ReLU, LeakyReLU) on model performance, providing both an educational tool and practical insights into neural network design and optimization.

## 2 Current Progress

This section outlines the progress made so far in the implementation of our neural network for binary classification.

1. **Neural Network Implementation**
   (a) We have developed a fully connected neural network from scratch, consisting of an input layer, two hidden layers (Hidden Layer 1 with 128 neurons and hidden layer 2 with 64 neurons for most of our datasets), and an output layer, designed for binary classification tasks. The model has been trained for 1000 epochs with a learning rate of 0.01 and a fixed random seed of 3.
   (b) We have implemented key functionalities: Forward propagation (calculates activations), Backward propagation (updates weights using gradient descent & different backpropagation techniques were applied based on the activation functions), Loss function: Log loss computation, and Parameter initialization & updates.
   (c) We have defined a *neural_network* function that integrates all components and saves the best model along with the error history.
2. **Dataset Preparation**
   We have loaded the MNIST, FashionMNIST, Iris, Dexter, Madelone, Spambase, and Leukemia datasets, preprocessed them for binary classification (labels 0 and 1), and converted them into NumPy arrays.
3. **Model Training and Evaluation**
   We have trained models using Sigmoid, ReLU, and LeakyReLU activation functions.
4. **Accuracy Evaluation**
   We have implemented train and test accuracy calculations across all datasets and executed them using all three activation functions. Figure 1 shows the loss curves for MNIST, FashionMNIST, and Iris, and Figure 2 presents the accuracy statistics.




(a) (b)

(c)

Figure 1: Comparison of Training Loss for (a)MNIST, (b)F-MNIST and (c)Iris Datasets using Sigmoid, ReLU, and LeakyReLU Activation Functions.

## 3 Challenges

Building a neural network from scratch for binary classification poses key challenges in design, configuration, and initialization, affecting performance on all datasets.

1. Implementing a neural network without pre-built libraries demands a deep understanding of its mathematical foundations. Correctly coding forward propagation, backward propagation with gradient descent, and activation functions (Sigmoid, ReLU, LeakyReLU) has been challenging.
2. **Determining Optimal Hyperparameters:** A key challenge for us was determining the optimal set of hyperparameters to achieve effective model performance. This included: Number of hidden layers (currently 2 with 128 and 64 neurons), Number of neurons in each layer, Number of iterations (currently 1000), and Learning rate (currently 0.01).
3. **Defining Random Seed:** Setting seed (currently 3) for reproducibility.
4. **Weight Initialization:** Weight initialization for ReLU and LeakyReLU activation functions remains an unresolved challenge.

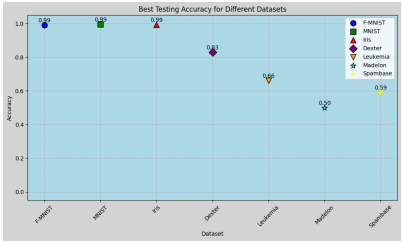



Figure 2: Best Testing Accuracy for Different Datasets