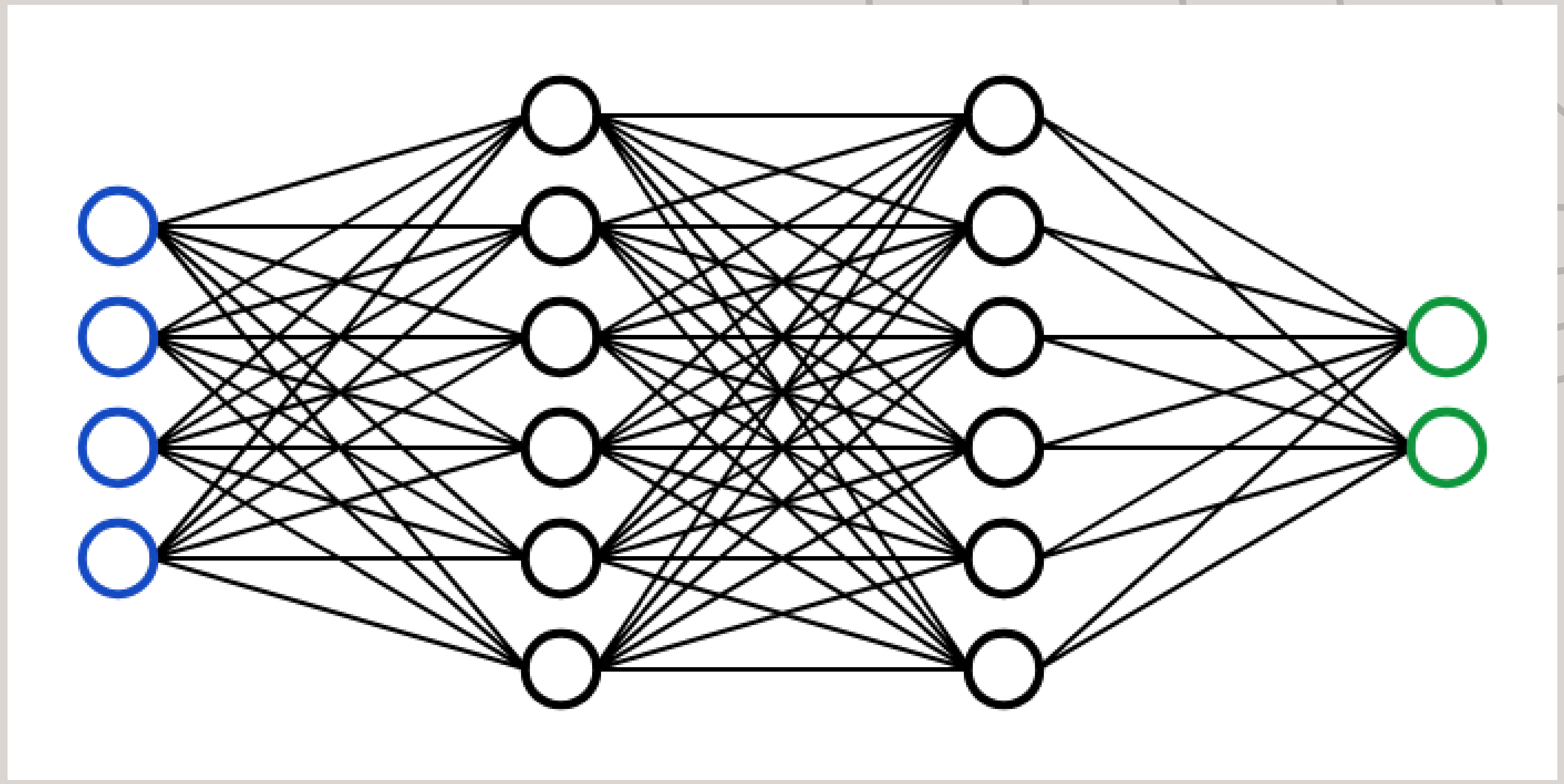
Implementation of Multilayer Perceptron Without Using Libraries

Abstract

This project implements a Multilayer Perceptron (MLP) from scratch in MATLAB to classify student performance based on factors like study time and previous scores. Key components such as forward propagation, backward propagation, and gradient descent were built without external libraries. The MLP successfully learned to predict performance, demonstrating the practicality of constructing neural networks from basic principles and their effectiveness in solving real-world problems.

What Is a Multilayer Perceptron (MLP)?

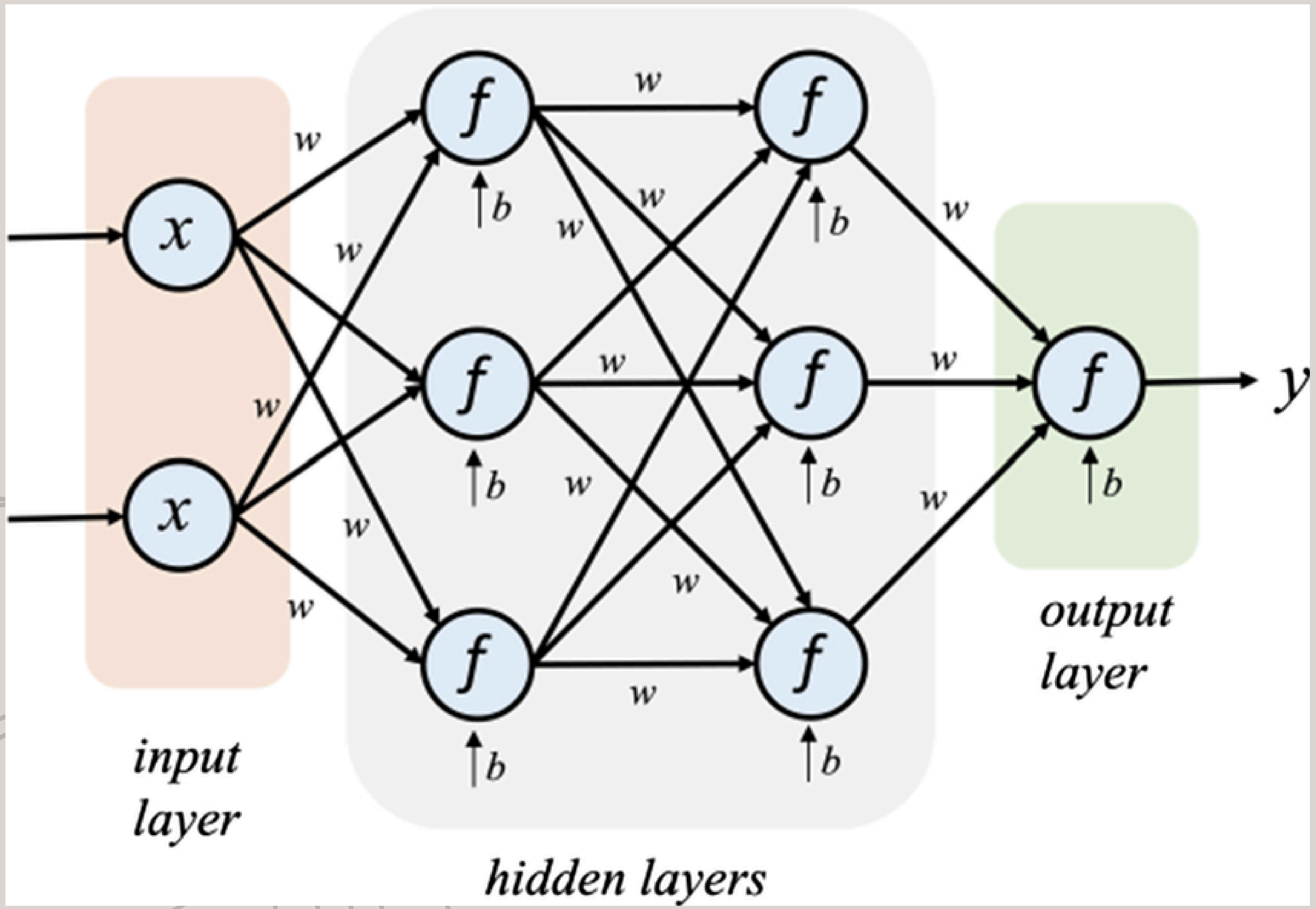


It's a type of artificial neural network that mimics how human brain learn.

It takes input data, processes it through layers, and makes a prediction.

Applications of MLP: Image and speech recognition, predictive analytics...

MLP Architecture



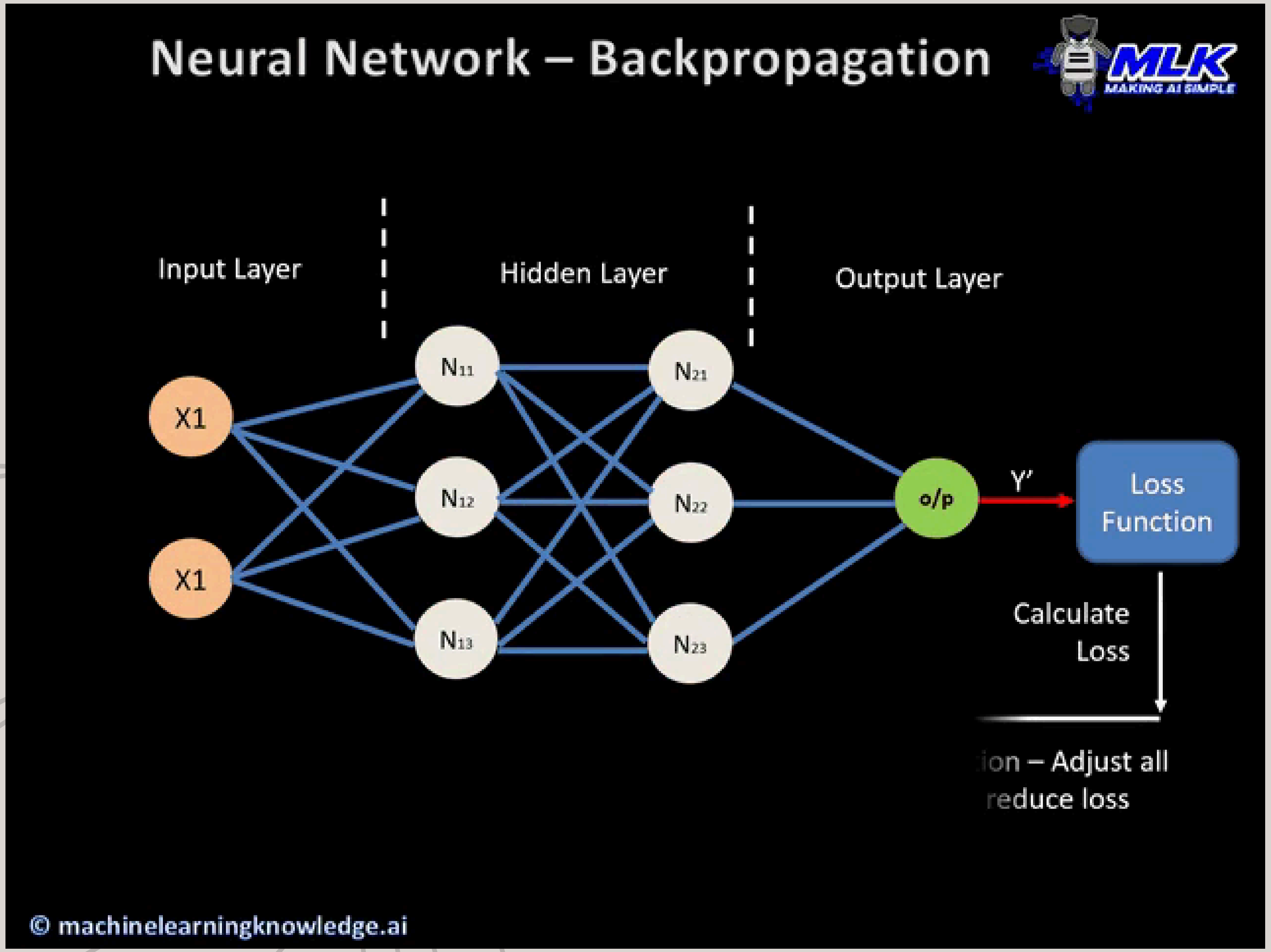
Components

Input Layer
Accepts input features
Hidden Layer
Performs transformations
Output Layer
Generates predictions

Key Features

Nurons, weights and biases
Activation functions

Steps in Implementation



1) Forward Propagation
- Compute weighted sum
- Apply activation function

2) Backward Propagation
- Calculate loss and gradients
- Update weights using gradient descent

3) Iterate Until Convergence
- Repeat steps to minimize the loss function.

Hyperparamethers

Hyperparameters are parameters used to control the training process of a machine learning model. Unlike model parameters, which are learned during training (e.g., weights in a neural network), hyperparameters are set before the training begins and are not updated during the process.

Model Hyperparameters

- Number of layers in network
- Nuber of neurons per layer
- Type of activation function

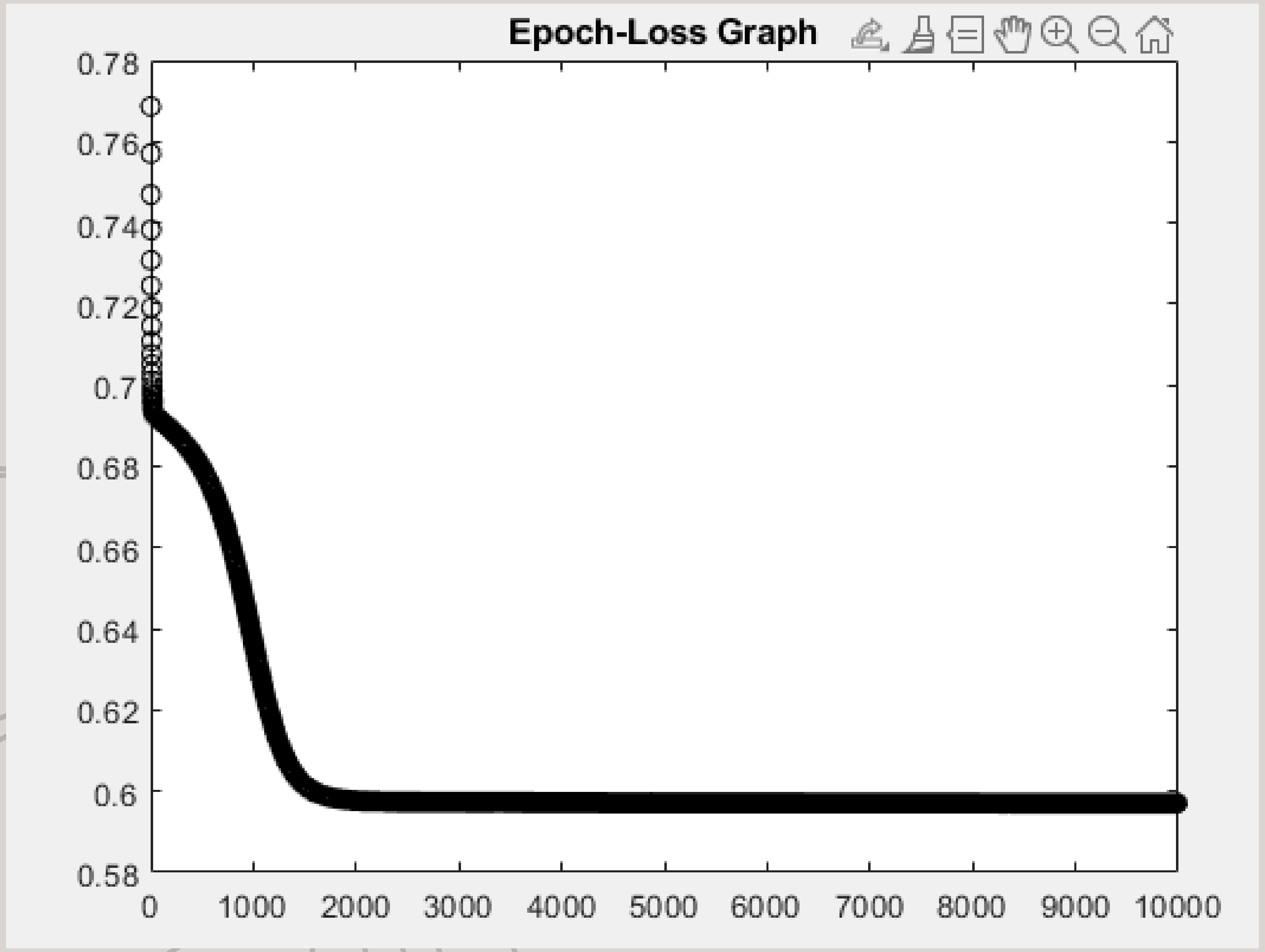
Training Hyperparameters

- Learning rate
- Batch size
- Numer of training epchs
- Optimization algorithm
- Implementation

MATLAB Implementation

1) Initialize weights and biases.
2) Define activation and loss functions
3) Implement forward propagation and compute outputs
4) Implement backpropagation and update weights
5) Train the model iteratively
6) Test model on seperate data and calculate accuracy

Result



Time: 15.74s
Accuracy: %96.90

Configuration:

- Dataset train size ratio: 0.8
- Data Normalization: Min-Max
- Structure: [5, 10, 5, 1]
- Activation Function: sigmoid
- Learning rate: 0.0001
- Epochs: 10,000

Challenges and Learnings

Challenges

- Debugging gradient computations
- Choosing the correct hyperparameters
- Ensuring convergence of the model

Key Learnings

- Hands-on experience with neural network mechanics
- Insights into optimization and activation functions

Conclusion

Key Points Recap

- MLP is a foundational neural network model
- Manual implementation deepens understanding of core concepts

Next Steps

- Explore more complex architectures like CNNs or RNNs
- Optimize the code for efficiency

Thank You

You can find my code files and this presentation on my GitHub

github.com/alibaransel/neural-network-matlab-no-lib