



---

# FINAL PROJECT PART 1

---

Computational Intelligence Project



Name	ID
Amr Khaled Abd Elsadik	2100524
Arwa Ramadan Abdelfattah	2100647
Hla Ehab Fawzy	2100708
Mario Sameh Samir	2100364
Youssef Ahmed Abd-El-Fattah	2101059



## Contents

FINAL PROJECT PART 1 .....	1
Introduction.....	2
library design and architecture choices.....	2
Results from the XOR test.....	2

# Introduction

This project focuses on building a complete neural network library from scratch using Python and NumPy, with the goal of understanding and implementing the core mechanisms of deep learning without relying on external frameworks. The library includes modular components for layers, activation functions, loss functions, and optimization, all integrated through a simple network structure designed for easy testing and extension.

## library design and architecture choices

The library is structured in a modular and extensible manner, enabling individual components to be easily modified, tested, or expanded. Core modules include:

- **layers.py**: Implements a base Layer class and a Dense layer with weight, bias, and gradient management.
- **activations.py**: Provides activation layers such as ReLU, Sigmoid, Tanh, and Softmax, each implemented as differentiable layers.
- **losses.py**: Contains the Mean Squared Error (MSE) loss, including both the forward computation and the initial gradient w.r.t. predictions.
- **optimizer.py**: Implements a simple yet effective Stochastic Gradient Descent optimizer for parameter updates.
- **network.py**: Defines a Sequential-style class that orchestrates the full forward and backward passes through the network.

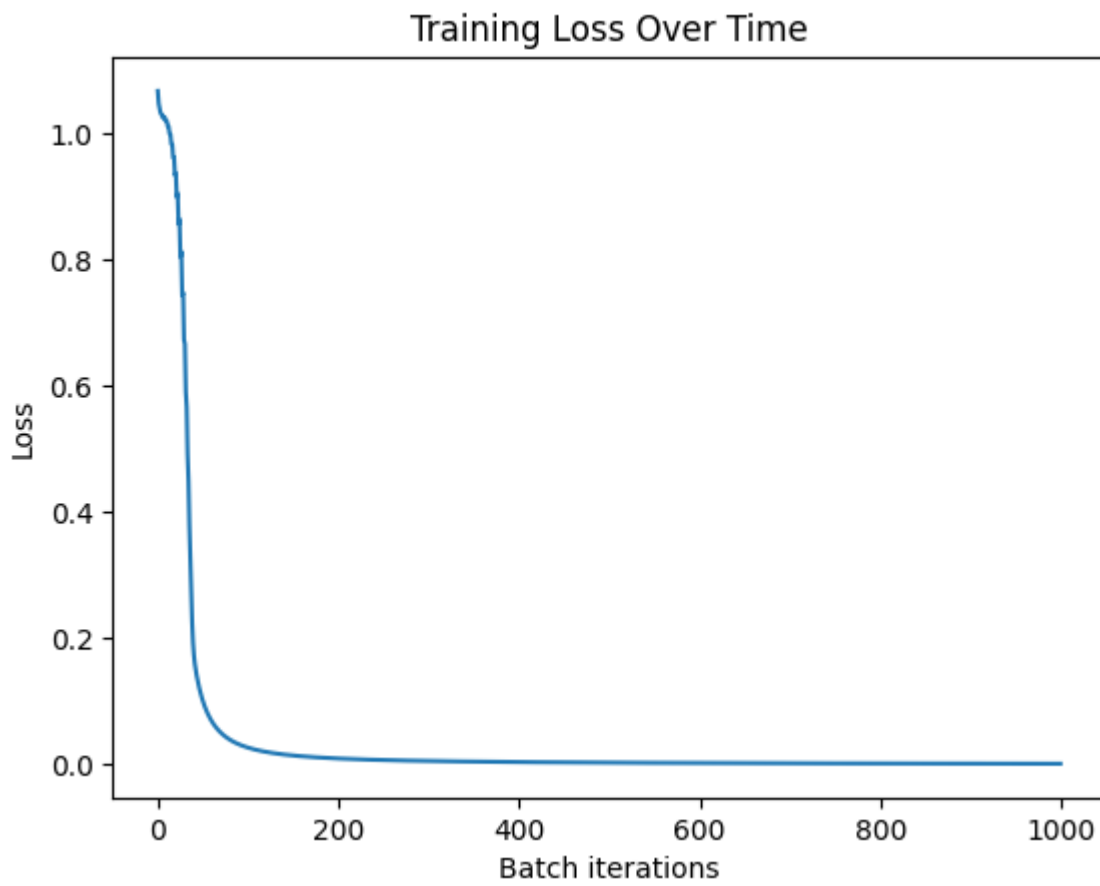
This modularity ensures clarity, flexibility, and easy debugging, while closely mirroring the design philosophy of professional deep learning libraries. Each layer encapsulates its own forward and backward logic, and the Network class integrates them into a cohesive training pipeline.

## Results from the XOR test

After training, the network successfully classified all four XOR input combinations with high accuracy. The results confirm that the library correctly performs forward propagation, computes gradients, and updates parameters using backpropagation. This initial validation provides a strong foundation for the more complex autoencoder and classification tasks that follow in subsequent sections.

```
predictions: [[-0.96409383]
 [ 0.96353676]
 [ 0.96015632]
 [-0.96415383]]
```

*Figure 1: the result of XOR test*



*Figure 2: Loss vs iterations*