Basic Of Deep Learning - Mid Semester Project Part 1

Amos Zohar (ID. 311402812), Gal Havshush (ID. 207389909)

Submitted as mid-semester project report for Basic Of Deep Learning course, Colman, 2024

1 Introduction

Deep learning has emerged as a transformative paradigm in artificial intelligence, offering sophisticated tools for solving complex challenges across domains such as computer vision, natural language processing, and robotics. Fundamentally, deep learning employs neural networks as mathematical models to discern intricate patterns within complex datasets. This project focuses on the manual implementation of a neural network using NumPy to classify hand-signed digits, providing an in-depth exploration of the theoretical and practical facets of neural network design. By eschewing high-level frameworks, the project facilitates a deeper understanding of the mechanics of forward propagation, backpropagation, and optimization. The dataset employed comprises grayscale images representing digits 0 through 9. To simplify the task and emphasize foundational principles, the study narrows its focus to binary classification of two selected classes. This approach provides a robust framework for investigating critical components such as data preprocessing, model architecture, and optimization strategies. Ultimately, the project highlights the effectiveness of simple neural networks while establishing a basis for more advanced machine learning models.

1.1 Data

Dataset: The Sign Language Digits dataset comprises 5,000 grayscale images, each with a resolution of 28x28 pixels. This curated dataset is well-suited for foundational experiments in image classification.

Preprocessing:

- Pixel intensities were normalized to the range [0, 1], enhancing numerical stability and facilitating efficient gradient-based optimization.
- Two classes, 4 and 9, were selected for binary classification and mapped to labels 0 and 1, respectively.

- The dataset was divided into training (80%) and testing (20%) subsets, ensuring representative distributions for both.

Filtered Data Statistics:

- Training set: 800 samples evenly divided between the two classes.
- Test set: 200 samples evenly divided.
- Balanced subsets ensure unbiased performance evaluation.

1.2 Problem

This project focuses on classifying hand-signed gestures into two categories: 4 (label 0) and 9 (label 1). The goal is to develop a neural network capable of accurately distinguishing between these two classes using a dataset of grayscale images. The training process was conducted entirely without the aid of any deep learning framework, relying exclusively on NumPy for all computational tasks, including forward propagation, backpropagation, and optimization. By adopting this approach, the project aims to provide a deeper understanding of neural network mechanics and the nuances of implementing these systems manually.

2 Solution

2.1 General approach

The proposed solution involves a feedforward neural network with a single hidden layer, implemented entirely in NumPy. The architecture was kept intentionally simple to match the straightforward nature of the binary classification task. The sigmoid activation function was used along with the Binary Cross-Entropy (BCE) loss function, which effectively quantifies prediction error in binary classification tasks. Gradient descent optimizes the network by iteratively minimizing BCE loss and refining model weights.

2.2 Design

The model employs a straightforward feedforward neural network with a single hidden layer, reflecting the simplicity of the binary classification task. This architecture balances computational efficiency and learning capacity, allowing effective training on the provided dataset.

Implementation Details:
The training process completed within a few seconds. Challenges included addressing numerical stability during Binary Cross-Entropy (BCE) loss calculations and ensuring proper gradient convergence through careful hyperparameter tuning.

Algorithm 1 Gradient Descent for Neural Networks

Initialize weights W and biases b randomly
for each epoch in the range of total epochs **do**
  for each training sample (x, y) **do**
    Forward Pass:
    Compute predictions $z = Wx + b$
    Compute activation $a = \sigma(z)$ where σ is the activation function
    Compute loss $L = BCE(a, y)$
    Backward Pass:
    Compute gradients $\frac{\partial L}{\partial W}$ and $\frac{\partial L}{\partial b}$
    Update Parameters:
    $W \leftarrow W - \eta \frac{\partial L}{\partial W}$ ▷ η is the learning rate
    $b \leftarrow b - \eta \frac{\partial L}{\partial b}$
  end for
end for

This iterative process ensures that the network minimizes the loss function while progressively refining its predictions for the classification task.

Model Architecture:

- **Input Layer**: 784 neurons, corresponding to the flattened 28x28 input grid.
- **Hidden Layer**: 10 neurons with sigmoid activation, facilitating non-linear transformations.
- **Output Layer**: A single neuron outputs the probability of class membership, with a decision threshold of 0.5.
- **Loss Function**: BCE loss penalizes incorrect predictions logarithmically, aligning with theoretical principles.
- **Learning Rate**: A value of 0.1 was selected to balance convergence speed and stability.
- **Epochs**: The model was trained over 10 epochs, providing ample opportunity for learning.

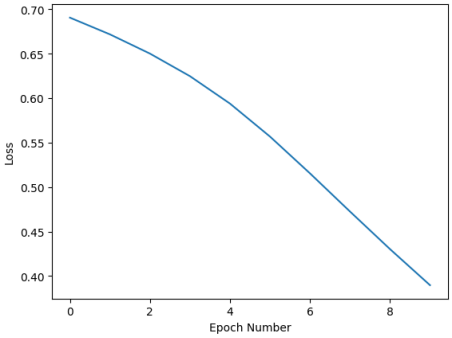
3 Base Model

The base model explores key hyperparameters, including the number of epochs, neurons in the hidden layer, and learning rate, to understand their impact on accuracy, loss reduction, and convergence speed. By systematically varying these parameters, the experiments reveal how each contributes to optimizing model performance.

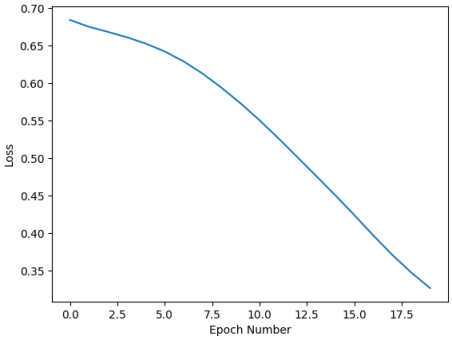
3.1 Experiments and Results

Number of Epochs: Using 1 neuron in the hidden layer and a learning rate of 0.01:

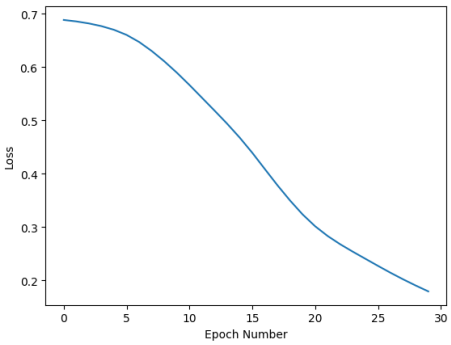
- 10 epochs: Initial Loss = 0.6906, Final Loss = 0.3897, Training Time = 913 ms, Accuracy = 88.5%.



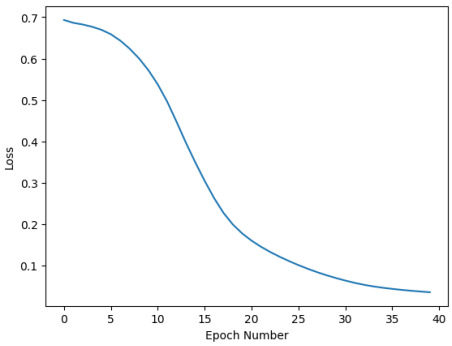
- 20 epochs: Initial Loss = 0.6843, Final Loss = 0.3266, Training Time = 1.18 s, Accuracy = 90.5%.



- 30 epochs: Initial Loss = 0.6886, Final Loss = 0.1794, Training Time = 2.15 s, Accuracy = 95.5%.



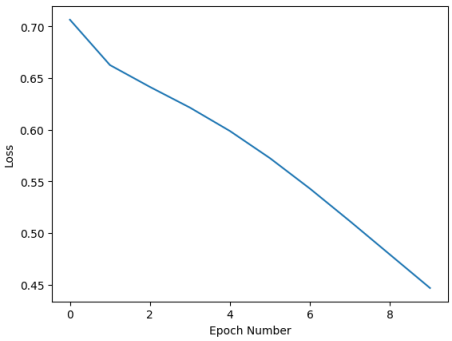
- 40 epochs: Initial Loss = 0.6932, Final Loss = 0.0356, Training Time = 3.04 s, Accuracy = 100%.



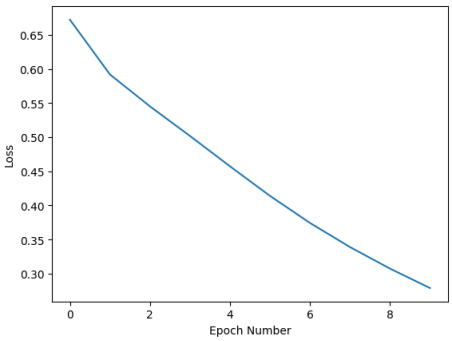
These results highlight the importance of sufficient training iterations to achieve optimal accuracy and convergence.

Number of Neurons in the Hidden Layer: Using a learning rate of 0.01 and 10 epochs:

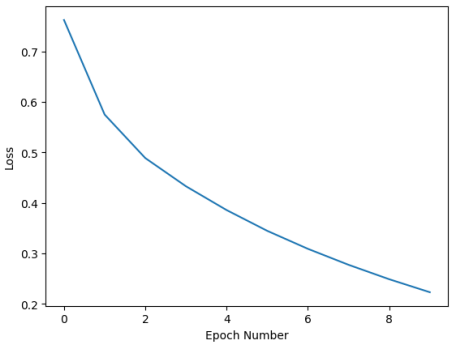
- 5 neurons: Initial Loss = 0.7066, Final Loss = 0.4467, Training Time = 689 ms, Accuracy = 84.5%.



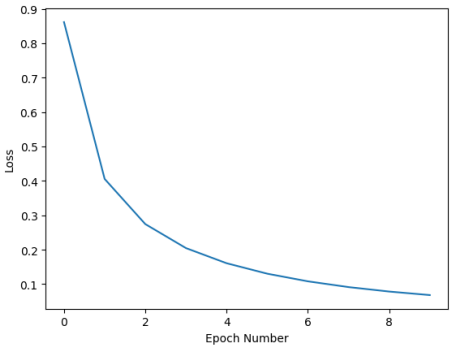
- 10 neurons: Initial Loss = 0.6723, Final Loss = 0.2788, Training Time = 701 ms, Accuracy = 91.5%.



- 20 neurons: Initial Loss = 0.7625, Final Loss = 0.2230, Training Time = 1.45 s, Accuracy = 96.0%.



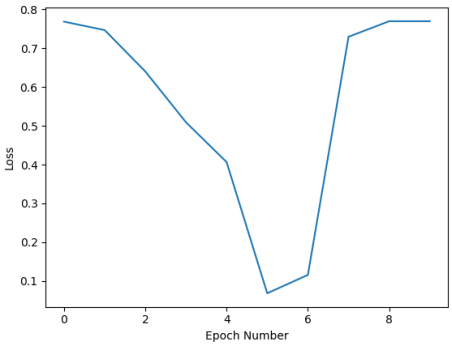
- 60 neurons: Initial Loss = 0.8614, Final Loss = 0.0682, Training Time = 2.68 s, Accuracy = 97.5%.



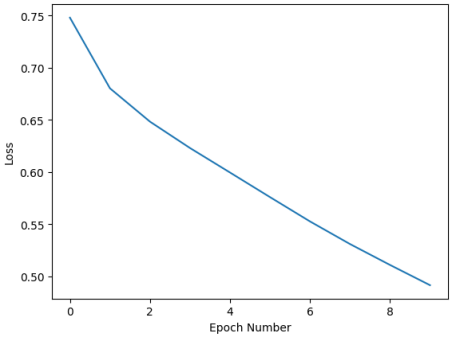
Adding neurons increased accuracy and reduced loss but required more computational resources.

Learning Rate: Using 1 neuron in the hidden layer and 10 epochs:

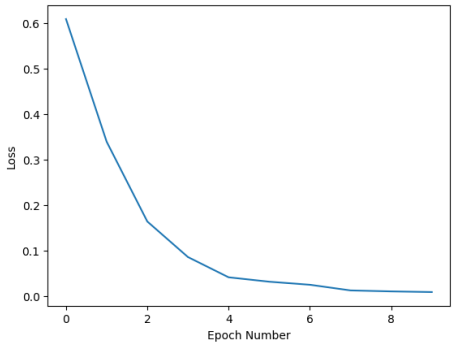
- Learning rate = 1: Initial Loss = 0.7688, Final Loss = 0.7699, Training Time = 636 ms, Accuracy = 50%.



- Learning rate = 0.01: Initial Loss = 0.7479, Final Loss = 0.4915, Training Time = 599 ms, Accuracy = 80.5%.



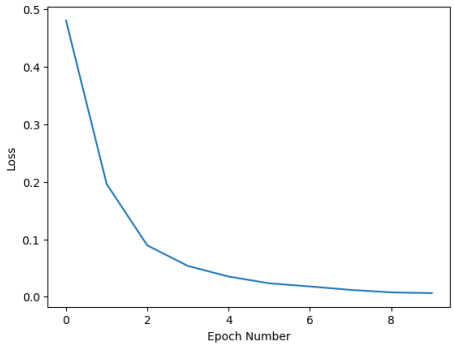
- Learning rate = 0.1: Initial Loss = 0.6085, Final Loss = 0.0090, Training Time = 668 ms, Accuracy = 100%.



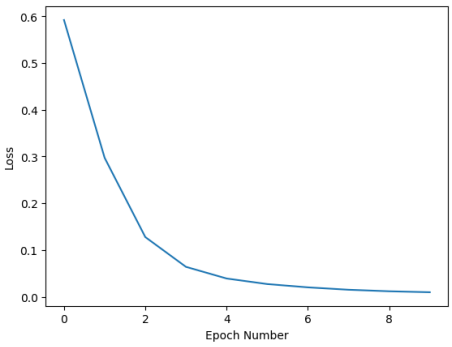
A learning rate of 0.1 was optimal for this task.

The hyperparameters chosen for the final model were: Number of Epochs: 10, Neurons in Hidden Layer: 10, Learning Rate: 0.1.
To validate these choices, additional tests were performed using the selected configuration. These tests demonstrated consistent performance across multiple runs, reinforcing the robustness of the chosen parameters:

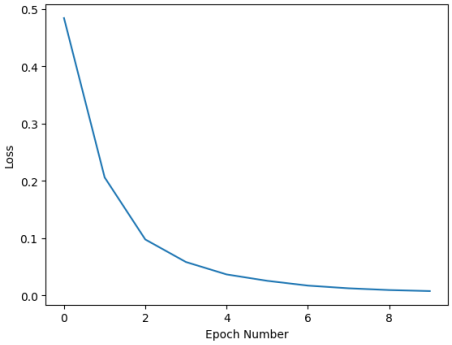
- **Test 1**: Initial Loss: 0.4803, Final Loss: 0.0063, Training Time: 704 ms, Accuracy: 100%.



- **Test 2**: Initial Loss: 0.5919, Final Loss: 0.0097, Training Time: 757 ms, Accuracy: 99.5%.



- **Test 3**: Initial Loss: 0.4841, Final Loss: 0.0074, Training Time: 740 ms, Accuracy: 100%.



4 Discussion

The experiments conducted demonstrated how hyperparameter selection profoundly influences neural network performance. Increasing the number of epochs allowed the network to converge effectively, while additional neurons in the hidden layer enhanced its capacity to model complex relationships. The choice of learning rate proved critical, with a value of 0.1 yielding the best balance between convergence speed and accuracy. The results show that iterative experimentation is key, as larger networks and longer training sessions improved accuracy but also raised computational costs. This project highlights the effectiveness of basic neural network architectures for binary classification, achieving perfect test set accuracy due to careful design and tuning of key components. Addressing challenges like numerical instability and gradient consistency proved crucial, underscoring the value of precise implementation and hyperparameter optimization.

Potential enhancements for future research include:

- Extending the architecture to multi-class classification using softmax activation.
- Investigating alternative activation functions, such as ReLU to improve gradient behavior.
- Expanding the model's complexity with additional layers and neurons to handle more sophisticated datasets.
- Using a Convolutional Neural Network (CNN) to better capture spatial hierarchies in image data and improve classification performance.

5 Code

The complete implementation is accessible in a **Colab notebook**

References

- **Sign Language Digits Dataset**
- **Latex Algorithm Implementation**