Machine Learning Journal

Assessment Task 2 – Mitchell Kijurina – 14274334

Option 1: *study a fundamental machine learning model*

Table of Contents

[1. Introduction 4](#_Toc147771535)

[1.1 Purpose of project 4](#_Toc147771536)

[1.2 Objectives 4](#_Toc147771537)

[1.3 Scope and Limitations 4](#_Toc147771538)

[2. Background and Related Concepts 5](#_Toc147771539)

[2.1 What is a Neural Network? 5](#_Toc147771540)

[2.2 Introduction to Multi-Layer Perceptrons (MLPs) 5](#_Toc147771541)

[2.3 Overview of Activation Functions 6](#_Toc147771542)

[2.4 Understanding Loss Functions 6](#_Toc147771543)

[2.5 Basics of Training Algorithms 6](#_Toc147771544)

[3. Problem Description 7](#_Toc147771545)

[3.1 Dataset Overview (MNIST) 7](#_Toc147771546)

[3.2 Problem Statement 7](#_Toc147771547)

[4. Theoretical Foundations 8](#_Toc147771548)

[4.1 Neural Network Architecture 8](#_Toc147771549)

[4.2 Forward Propagation 8](#_Toc147771550)

[4.3 Backward Propagation 8](#_Toc147771551)

[4.4 Loss Function (Cross-Entropy) 8](#_Toc147771552)

[4.5 Gradient Descent Algorithm 8](#_Toc147771553)

[5. Implementation Details 9](#_Toc147771554)

[5.1 Setting up the Environment 9](#_Toc147771555)

[5.2 Data Preprocessing 9](#_Toc147771556)

[5.3 Coding the Neural Network 9](#_Toc147771557)

[5.3.1 Input and Hidden Layers 9](#_Toc147771558)

[5.3.2 Output Layer 9](#_Toc147771559)

[5.3.3 Activation Functions 9](#_Toc147771560)

[5.3.4 Loss Function 9](#_Toc147771561)

[5.3.5 Training Loop 9](#_Toc147771562)

[5.4 Hyperparameter Tuning 9](#_Toc147771563)

[6. Results and Discussion 10](#_Toc147771564)

[6.1 Model Performance Metrics 10](#_Toc147771565)

[6.2 Loss and Accuracy Plots 10](#_Toc147771566)

[6.3 Interpretation of Results 10](#_Toc147771567)

[7. Conclusion 11](#_Toc147771568)

[7.1 Summary of Findings 11](#_Toc147771569)

[7.2 Implications 11](#_Toc147771570)

[7.3 Future Work 11](#_Toc147771571)

[Sources 12](#_Toc147771572)

[ChatGPT logs 12](#_Toc147771573)

# 1. Introduction

## 1.1 Purpose of project

The primary purpose of this project is to facilitate a deep, hands-on understanding of the fundamental concepts that drive the functioning of Multi-Layer Perceptrons (MLPs), a type of artificial neural network. By implementing an MLP from scratch, this study aims to bridge the gap between abstract theoretical principles and their concrete realization in code. The project serves as a pedagogical exercise to explore key aspects of neural network learning, such as hypothesis space, learning algorithms, and loss function design, within a practical context. This project aims to provide insights into how MLPs learn to make predictions or decisions based on data, thereby acting as a learning guide to both the implementation and theory of neural networks.

## 1.2 Objectives

The main objectives of this study are to:

* To explore the theoretical fundamentals of MLPs
* To implement an MLP model from scratch to demonstrate a clear understanding of its technical aspects.

## 1.3 Scope and Limitations

The study will focus solely on the architecture, theory, and implementation of MLPs, particularly within the context of the learning theory framework. The primary dataset for practical implementation will be the MNIST dataset for digit recognition. However, the study will not delve into variations of neural networks like Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs).

# 2. Background and Related Concepts

This section will cover a high level overview of coming topics that are explored more in-depth when looking at the implementation of the program.

## 2.1 What is a Neural Network?

Neural networks, also known as artificial neural networks (ANNs) or simulated neural networks (SNNs), are crucial to machine learning and deep learning algorithms. Their structure is inspired by the human brain, mimicking the way biological neurons signal to each other [1].

The neuron is the fundamental unit of the neural network. Each neuron, or a node, has the following components [2].

* Inputs
* Weights
* Bias
* Activation Function

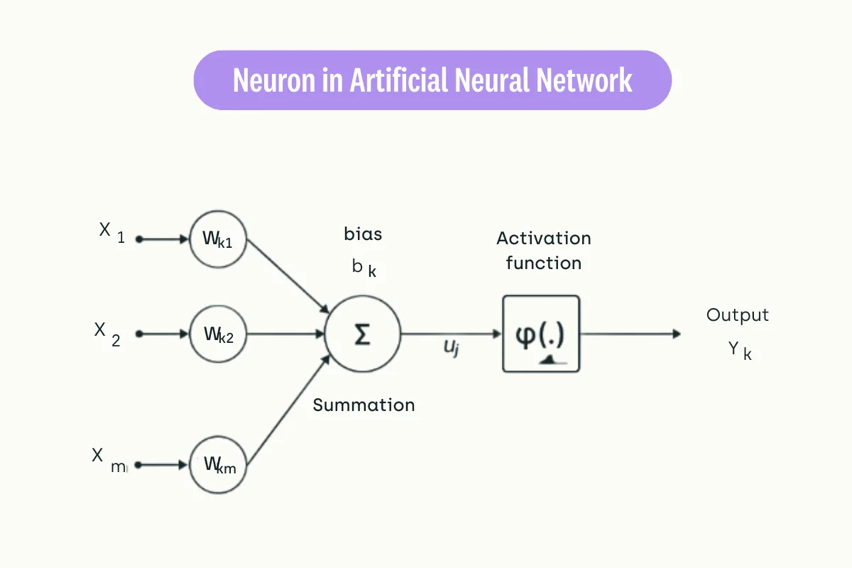


Figure - Diagram of a neuron [2]

## 2.2 Introduction to Multi-Layer Perceptrons (MLPs)

Multi-Layer Perceptrons (MLPs) are a type of feedforward neural network consisting of at least three layers: an input layer, one or more hidden layers, and an output layer. They can recognize more complex patterns with the inclusion of more layers and nodes per layer [2].

## 2.3 Overview of Activation Functions

Activation functions introduce non-linear properties to the network, enabling it to learn from the error and to handle complex problems. Common activation functions include Rectified Linear Unit (ReLU), Sigmoid, and Hyperbolic Tangent (tanh). These functions are crucial for neural network's learning process as they help in adjusting the weights during the training phase, aiding in the propagation of useful information through the network.

## 

## 2.4 Understanding Loss Functions

Loss functions are pivotal as they measure the disparity between the predicted values and actual values, guiding the optimization of the network's weights. The choice of a loss function, like Mean Squared Error (MSE) for regression tasks and Cross-Entropy Loss for classification tasks, can significantly affect the network's performance and convergence speed.

## 2.5 Basics of Training Algorithms

Training algorithms aim to minimise the loss function by iteratively adjusting the weights of the network. Backpropagation, combined with optimisation algorithms like Gradient Descent, is commonly used for this purpose. During training, the algorithm iteratively adjusts the weights to find the loss function's minimum, effectively "training" the network to improve its predictions over time.

# 3. Problem Description

## 3.1 Dataset Overview (MNIST)

The Modified National Institute of Standards and Technology (MNIST) dataset is one of the most well-known and widely used datasets in the machine learning community, particularly for benchmarking classification algorithms. It consists of 60,000 grayscale images for training and an additional 10,000 images for testing. Each image in the dataset is 28x28 pixels and represents a handwritten digit from 0 to 9.

**Characteristics of MNIST:**

* **Grayscale Images**: The images are grayscale, simplifying the complexity as compared to coloured images. Each pixel has a single intensity value between 0 (black) and 255 (white).
* **Uniform Size**: All images are resized to a uniform 28x28 pixel size, thereby eliminating the need for additional pre-processing steps related to image resizing.
* **Digit Labels**: Each image comes with a corresponding label indicating the actual digit it represents, making it a supervised learning problem.
* **Balanced Classes**: The dataset is relatively balanced, meaning that there are almost an equal number of images for each of the 10-digit classes.

**Relevance to the Project:**

The MNIST dataset provides an ideal ground for implementing and understanding the Multi-Layer Perceptron model. Its simplicity and well-defined structure allow for a focus on the learning algorithm rather than data pre-processing. Moreover, the nature of the task—digit recognition—serves as a practical yet challenging problem, aligning well with the objectives of dissecting the intricacies of neural network learning both in theory and in practice.

## 3.2 Problem Statement

# 4. Theoretical Foundations

## 4.1 Neural Network Architecture

## 4.2 Forward Propagation

## 4.3 Backward Propagation

## 4.4 Loss Function (Cross-Entropy)

## 4.5 Gradient Descent Algorithm

# 5. Implementation Details

## 5.1 Setting up the Environment

## 5.2 Data Preprocessing

## 5.3 Coding the Neural Network

### 5.3.1 Input and Hidden Layers

### 5.3.2 Output Layer

### 5.3.3 Activation Functions

### 5.3.4 Loss Function

### 5.3.5 Training Loop

## 5.4 Hyperparameter Tuning

# 6. Results and Discussion

## 6.1 Model Performance Metrics

## 6.2 Loss and Accuracy Plots

## 6.3 Interpretation of Results

# 7. Conclusion

## 7.1 Summary of Findings

## 7.2 Implications

## 7.3 Future Work

# Sources

* <https://www.ibm.com/topics/neural-networks#:~:text=Neural%20networks%2C%20also%20known%20as,neurons%20signal%20to%20one%20another>

## ChatGPT logs

* Code implementation chat