Lab Report 2

Understanding Artificial Neural Networks from Scratch

Course Code: COMP-341L

Course Name: Artificial Neural Networks Lab

Lab Number: 2

Lab Title: Understanding ANN from Scratch

Date: January 26, 2026

Name: Ali Hamza

Roll Number: B23F0063AI106

Section: B.S AI – Red

1 Objective

The objective of this lab is to understand the fundamental working principles of artificial neural networks by implementing a single neuron, analyzing different activation functions, studying the effect of bias on threshold behavior, exploring the necessity of hidden layers, and implementing softmax activation for multi-class classification.

2 Introduction

Artificial Neural Networks (ANNs) are computational systems inspired by the functioning of biological neurons. Rather than performing simple calculations, neurons act as decision-making units that determine whether incoming signals are significant enough to propagate forward.

This laboratory experiment focuses on building neural network components from scratch using Python, NumPy, and Matplotlib. The experiments simulate stress detection from speech features and aim to develop conceptual understanding of how neurons interpret signals, how activation functions shape decisions, and how network depth enables complex learning.

3 Prerequisites

- Python 3.8 or higher
- NumPy library
- Matplotlib library
- Basic knowledge of linear algebra and probability

4 Task 1: Building a Single Artificial Neuron

4.1 Objective

To design a neuron that makes decisions rather than performing only numerical computation, and to observe how different activation functions affect its output.

4.2 Input Features

Three speech-related features were selected:

- Speech Rate (0–1 scale)
- Pitch Variation (0–1 scale)
- Pause Duration (0–1 scale)

The simulated input values were:

$$x = [0.85,\ 0.72,\ 0.25]$$

4.3 Weight Assignment

$$w = [0.4,\ 0.5,\ 0.1]$$

Pitch variation was assigned the highest importance due to its strong correlation with stress.

4.4 Bias Selection

A bias value of -0.3 was used to control neuron sensitivity. A negative bias requires stronger evidence before activation.

4.5 Activation Functions

The weighted sum was computed as:

$$z = w \cdot x + b$$

The following activation functions were applied:

$$\text{Sigmoid}(z) = \frac{1}{1 + e^{-z}} \tag{1}$$

$$\text{Tanh}(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{2}$$

$$\text{ReLU}(z) = \max(0, z) \tag{3}$$

4.6 Code Implementation

```
weighted_sum = np.dot(inputs, weights) + bias


output_sigmoid = 1 / (1 + np.exp(-weighted_sum))
output_tanh = np.tanh(weighted_sum)
output_relu = np.maximum(0, weighted_sum)
```

4.7 Results

- Weighted Sum: 0.435
- Sigmoid Output: 0.6071
- Tanh Output: 0.4106
- ReLU Output: 0.4350

4.8 Analysis

Different activation functions interpret the same evidence differently. Sigmoid provides probabilistic confidence, tanh produces signed activation, and ReLU outputs raw activation strength.

5 Task 2: Threshold and Bias Experiment

5.1 Objective

To observe how bias controls neuron activation behavior while keeping inputs and weights fixed.

5.2 Method

Bias was varied from -3.0 to $+3.0$ while computing neuron outputs using sigmoid, tanh, and ReLU functions.

5.3 Implementation

```
bias_range = np.linspace(-3.0, 3.0, 300)
z_values = weighted_sum_no_bias + bias_range

sigmoid_outputs = sigmoid(z_values)
tanh_outputs = tanh(z_values)
relu_outputs = relu(z_values)
```

5.4 Observation

- Sigmoid activates gradually
- ReLU activates suddenly at $z = 0$
- Tanh transitions smoothly through a neutral zone

5.5 Key Insight

Bias determines *when* a neuron activates, while the activation function determines *how* it activates.

6 Task 3: Limitation of a Single Neuron

6.1 Problem Description

A single neuron was unable to distinguish between calm-fast speakers and stressed-slow speakers due to overlapping linear decision boundaries.

6.2 Hidden Layer Solution

A multi-layer perceptron with two hidden neurons was implemented to learn intermediate patterns.

6.3 Forward Pass

```
def mlp_forward_pass(inputs):
    z1 = np.dot(inputs, W_hidden_1) + b_hidden_1
    z2 = np.dot(inputs, W_hidden_2) + b_hidden_2

    a1 = sigmoid(z1)
    a2 = sigmoid(z2)

    z_out = np.dot([a1, a2], W_output) + b_output
    return sigmoid(z_out)
```

6.4 Discussion

Hidden layers enable feature transformation and non-linear decision boundaries, allowing neural networks to solve complex real-world problems.

7 Task 4: Softmax for Multi-Class Classification

7.1 Objective

To convert raw network outputs into probability distributions across multiple emotional states.

7.2 Softmax Equation

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

7.3 Implementation

```
def softmax(z):
    z_shifted = z - np.max(z)
    exp_z = np.exp(z_shifted)
    return exp_z / np.sum(exp_z)
```

7.4 Results

- Calm: 0.5889
- Anxious: 0.2925
- Stressed: 0.1186

7.5 Interpretation

Softmax enforces competition among classes, ensuring probabilities remain within $[0, 1]$ and sum to 1.

8 Results and Observations

- Activation functions produce distinct neuron behaviors
- Bias strongly influences activation sensitivity
- Hidden layers enable non-linear learning
- Softmax provides interpretable confidence scores

9 Conclusion

This laboratory experiment provided deep conceptual understanding of artificial neural networks. By implementing neurons from scratch, it became evident that neural networks function as layered decision systems rather than simple mathematical models.

The experiments demonstrated how activation functions shape decisions, how bias controls sensitivity, why depth is required for complex pattern recognition, and how softmax enables probabilistic multi-class reasoning. These concepts form the foundation of modern deep learning systems.

10 References

1. Lab Manual: Lab 02 – Understanding ANN from Scratch

2. NumPy Documentation: `https://numpy.org/doc/`

3. Matplotlib Documentation: `https://matplotlib.org/`

Prepared by: Ali Hamza

Roll Number: B23F0063AI106

Section: B.S AI – Red

Date: January 26, 2026