Lab Report 3: Activation Functions

Course Code: COMP-341L
Course Name: Artificial Neural Networks Lab
Lab Number: 3
Lab Title: Activation Functions
Date: February 2, 2026

Name: Ali Hamza
Roll Number: B23F0063AI106
Section: B.S AI - Red

Objective

To implement and visualize common activation functions including Sigmoid, ReLU, Tanh, and Leaky ReLU using Python, and to understand their behavior, advantages, and limitations through conceptual analysis.

Introduction

Activation functions play a vital role in neural networks by introducing non-linearity, which allows the network to learn complex patterns. Without activation functions, neural networks would behave like linear models regardless of depth. This lab focuses on implementing four commonly used activation functions and analyzing their graphical behavior and practical use cases.

Task 1: Implementation and Visualization of Activation Functions

1.1 Sigmoid Activation Function

Formula:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Code Description: The sigmoid function is implemented using NumPy to map input values into a range between 0 and 1. To prevent numerical overflow caused by large exponential values, the input is clipped within a safe range before computation. This ensures stable results during plotting and calculations.

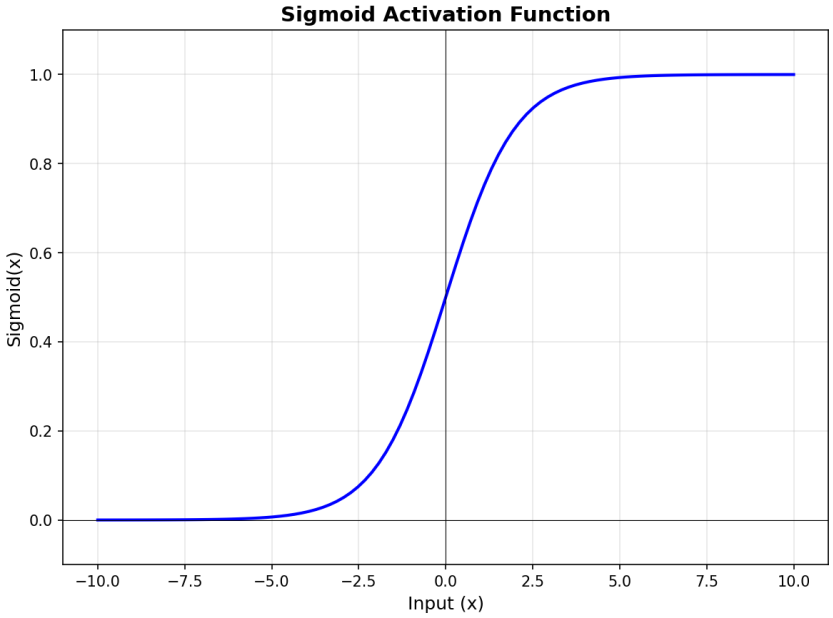

Figure 1: Sigmoid Activation Function

1.2 ReLU (Rectified Linear Unit)

Formula:

$$f(x) = \max(0, x)$$

Code Description: The ReLU function is implemented using NumPy's `maximum` function. It outputs zero for all negative input values and returns the input itself for positive values. This simple operation makes ReLU computationally efficient and suitable for deep neural networks.

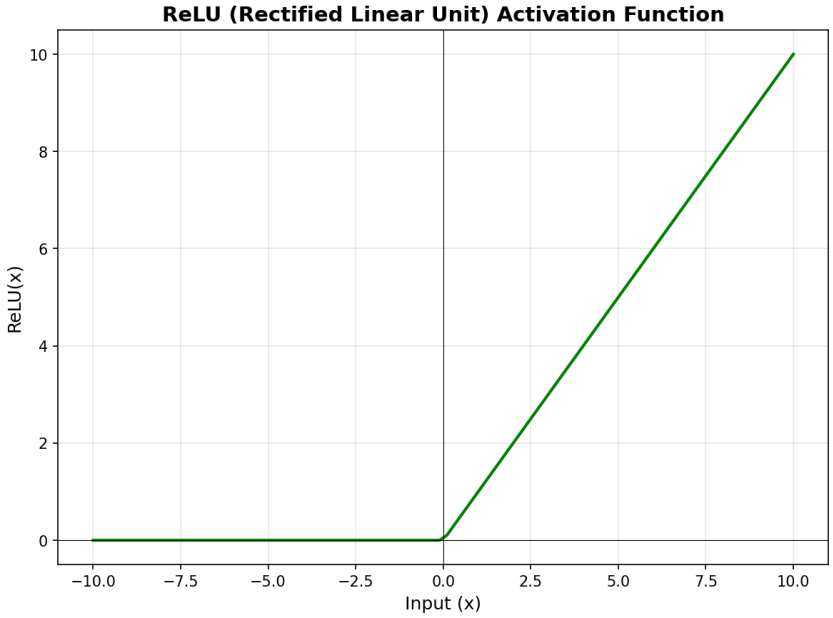

Figure 2: ReLU Activation Function

1.3 Tanh (Hyperbolic Tangent)

Formula:

$$f(x) = \tanh(x)$$

Code Description: The Tanh function is implemented using NumPy's built-in `tanh()` method. It converts input values into outputs ranging from -1 to 1. This zero-centered output helps improve convergence during training compared to sigmoid.

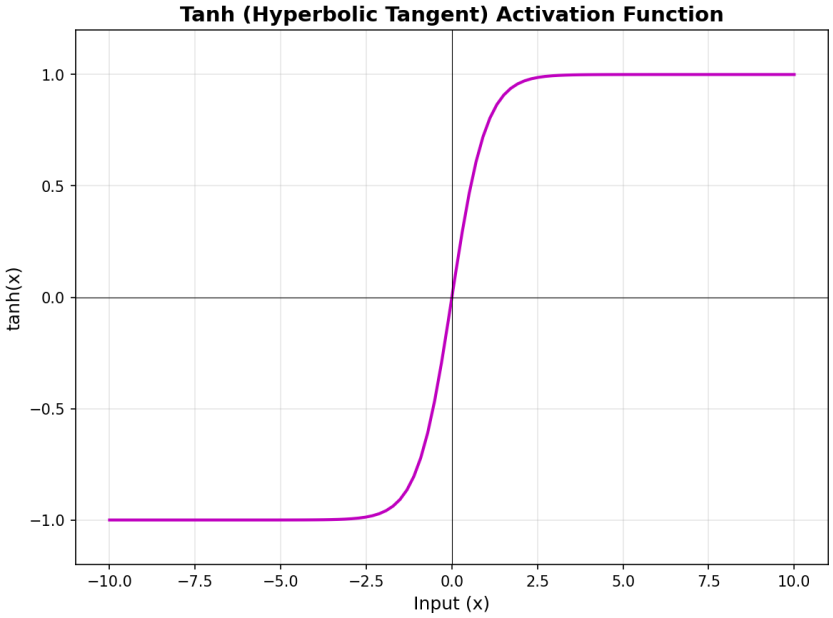

Figure 3: Tanh Activation Function

1.4 Leaky ReLU

Formula:

$$f(x) = \begin{cases} x, & x > 0 \\ \alpha x, & x \leq 0 \end{cases} \quad \text{where } \alpha = 0.01$$

Code Description: Leaky ReLU is implemented using NumPy's `where` function. For positive inputs, the function behaves like ReLU. For negative inputs, it allows a small slope controlled by α. This prevents neurons from becoming permanently inactive during training.

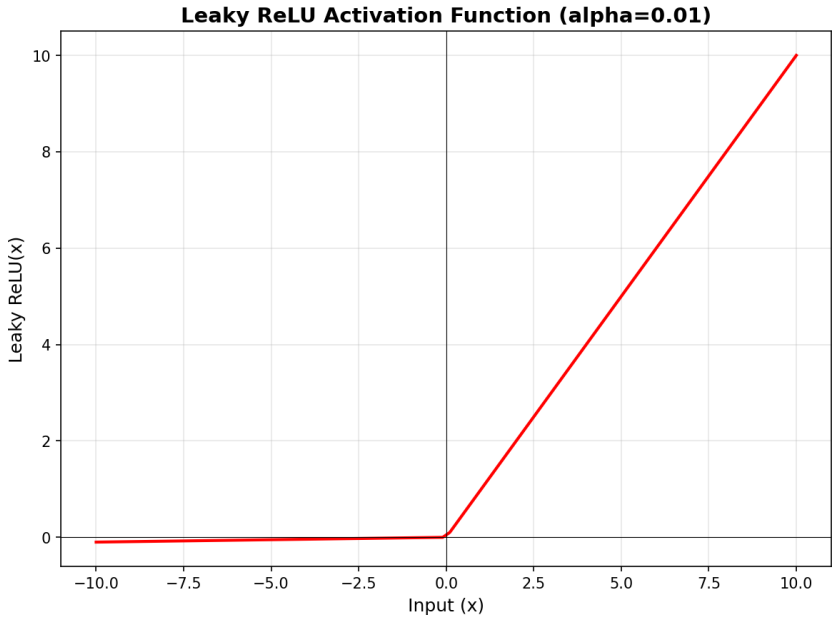

Figure 4: Leaky ReLU Activation Function

1.5 Comparison of All Activation Functions

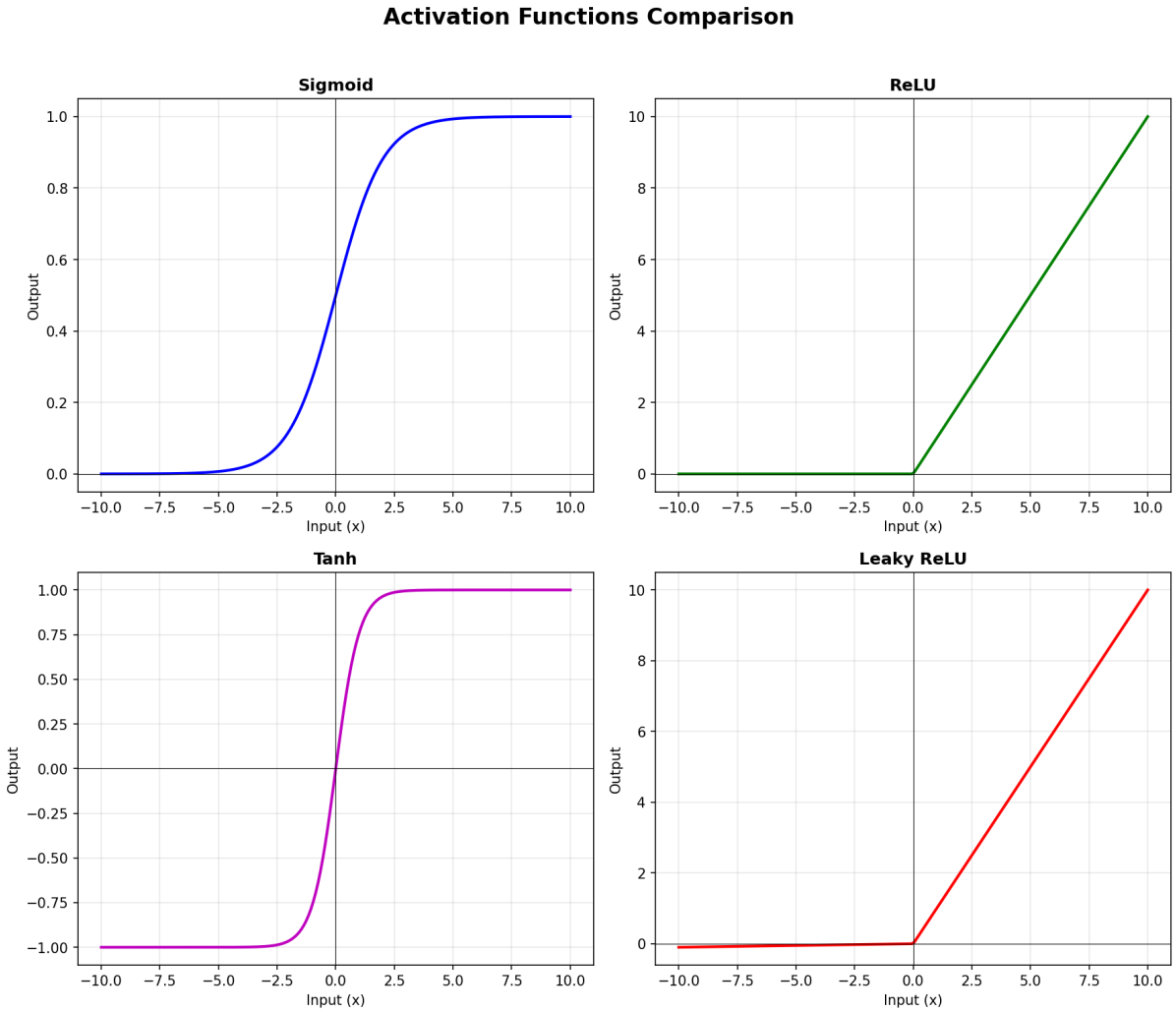

Figure 5: Comparison of Sigmoid, ReLU, Tanh, and Leaky ReLU

Task 2: Conceptual Questions

Q1: Which activation function is most suitable for hidden layers and why?

ReLU is most suitable for hidden layers because it is computationally efficient and helps reduce the vanishing gradient problem. It allows faster training and is widely used in deep neural networks.

Q2: What is the vanishing gradient problem and which activation functions suffer from it?

The vanishing gradient problem occurs when gradients become very small during back-propagation, causing early layers to learn slowly. Sigmoid and Tanh suffer from this issue due to saturation at extreme values, while ReLU and Leaky ReLU help mitigate it.

Q3: What issue does ReLU face and how does Leaky ReLU solve it?

ReLU can suffer from the dying ReLU problem, where neurons stop updating if they only receive negative inputs. Leaky ReLU solves this by allowing a small non-zero gradient for negative values.

Q4: Why is Softmax preferred in output layers for classification?

Softmax converts network outputs into probabilities that sum to one, making it ideal for multi-class classification problems. It also works well with cross-entropy loss.

Results and Observations

The plots clearly demonstrate how each activation function behaves for different input values. ReLU and Leaky ReLU show better gradient flow compared to Sigmoid and Tanh. The comparison plot highlights the differences in output range and saturation behavior.

Conclusion

In this lab, I learned how different activation functions influence the learning process of neural networks. By implementing and visualizing Sigmoid, ReLU, Tanh, and Leaky ReLU, I gained a clear understanding of their strengths and weaknesses.

I learned that ReLU is the preferred choice for hidden layers because it improves training efficiency, while Leaky ReLU helps avoid inactive neurons. Sigmoid and Tanh are useful in specific scenarios but are not ideal for deep networks. Overall, this lab improved my understanding of neural network design and the importance of selecting appropriate activation functions.

References

1. Lab Manual: Lab 03 - Activation Functions

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