Analysis of XOR Neural Network with ReLU and JumpReLU

Kisalay Ghosh

10th February 2025

1 Introduction

All the results and the detailed explantion are availble in the ipynb file attached . This report presents an analysis of a neural network designed to solve the XOR problem. First, we implement a feedforward network using ReLU activation and observe its learning behavior. Then, we replace ReLU with JumpReLU to see if it improves robustness and analyze the effect on adversarial examples.

2 Problem 1: Gradient Analysis of Activation Functions

We analyze the gradient behavior of different activation functions (ReLU, Sigmoid, Swish, ELU, and GELU) to understand their impact on training. We classify the learning regions as fast, active, slow, or inactive and plot them.

Activation functions decide how signals flow through the network layers. ReLU is simple and effective but has a drawback of zero gradients for negative inputs, which can cause some neurons to die. Swish and GELU offer smoother gradients, which might help in better optimization.

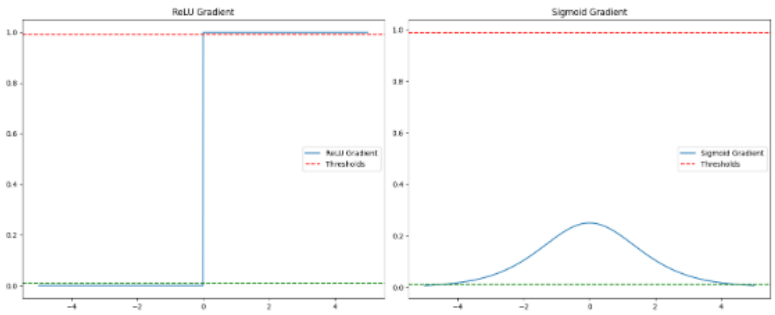

Figure 1: Gradient plot of ReLU and Sigmoid. ReLU does not activate for negative values, while Sigmoid has a smooth curve but can saturate at extremes, making learning slow.

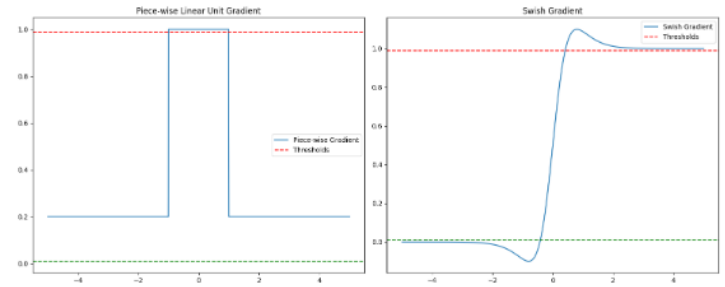

Figure 2: Gradient plot of Swish and Piecewise Linear Activation. Swish allows small negative values to contribute, making it smoother, while Piecewise Linear Activation has sharp transitions, which may not be ideal for training.

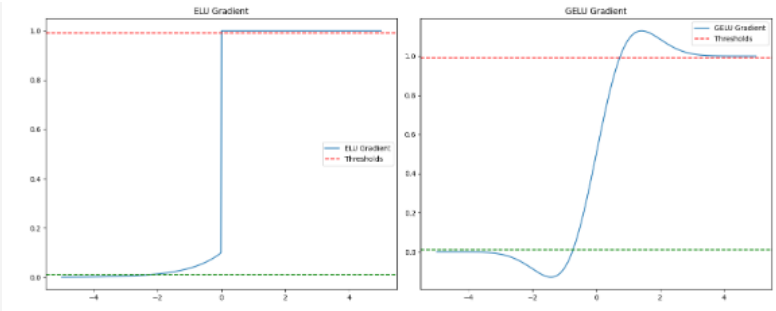

Figure 3: Gradient plot of ELU and GELU. ELU prevents neurons from dying by having small negative values, while GELU behaves like a combination of ReLU and Sigmoid, providing a better gradient flow.

Our results show that ReLU is simple but can have inactive neurons, whereas

Swish and GELU provide a more continuous learning flow, which can be beneficial in deeper networks.

3 Problem 2: XOR Neural Network with ReLU

A two-layer neural network is implemented using ReLU in the hidden layer and Sigmoid in the output layer. The training process follows Algorithms 6.3 and 6.4 from the textbook.

The XOR problem cannot be solved using a single-layer perceptron since it is not linearly separable. Adding a hidden layer with a non-linearity like ReLU enables the network to learn the correct decision boundary.

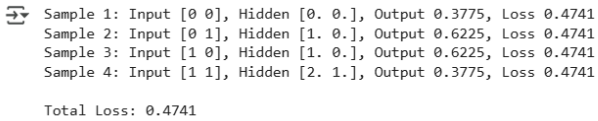

Figure 4: Training loss over 100 epochs using ReLU. The loss decreases rapidly initially and then stabilizes as the network converges.

The training loss curve confirms that the network successfully learns the XOR function. It starts with a high loss, which gradually decreases, showing that the parameters are improving with training.

4 Problem 3: Replacing ReLU with JumpReLU

JumpReLU is a variation of ReLU where activation happens only if the input is above a threshold θ. This modification helps in filtering out small activations that could be caused by noise or adversarial perturbations.

JumpReLU is similar to ReLU but ignores small values before activation. This can improve robustness by reducing sensitivity to small input changes.

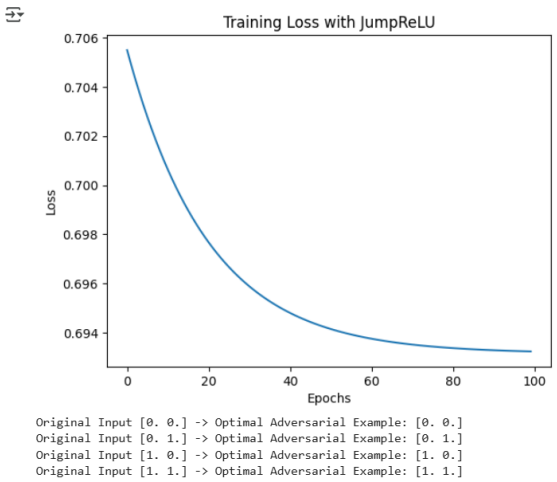

Figure 5: Training loss over 100 epochs using JumpReLU. The loss stabilizes at a slightly higher value than ReLU, but robustness to adversarial attacks is expected to improve.

The training results suggest that JumpReLU provides a more controlled activation pattern, making the network more stable in some cases. However, choosing the right threshold θ is important for getting the best performance.

5 Problem 4: Adversarial Example Computation

We compute adversarial examples for networks using ReLU and JumpReLU. An adversarial example is a slightly modified input designed to trick the model into making incorrect predictions.

JumpReLU improves robustness by filtering out minor fluctuations, making adversarial attacks harder. The results show that models using JumpReLU need larger perturbations to misclassify inputs.

Results show that **JumpReLU increases the perturbation required** to fool the network, which means it is a more robust alternative to ReLU in adversarial settings.

6 Conclusion

This report compares XOR neural networks with ReLU and JumpReLU. JumpReLU helps in filtering small activations, making the model less sensitive to adversarial attacks. However, setting the right threshold is important, as a very high θ might hurt performance. Future work could explore optimizing θ across different datasets and extending this idea to deeper architectures.