

Analysis of 5 Common Activation Functions in Deep Learning

Introduction to control and machine learning, SS25

Haider Ali, Muhammad Sulayman Khan, Rahul Sawhney, Kateryna Kolesova

-
- 01** Introduction
 - 02** Sigmoid
 - 03** Tangent
 - 04** ReLU
 - 05** Leaky ReLU
 - 06** SoftMax
 - 07** Simulation/Visualisation
 - 08** Conclusion

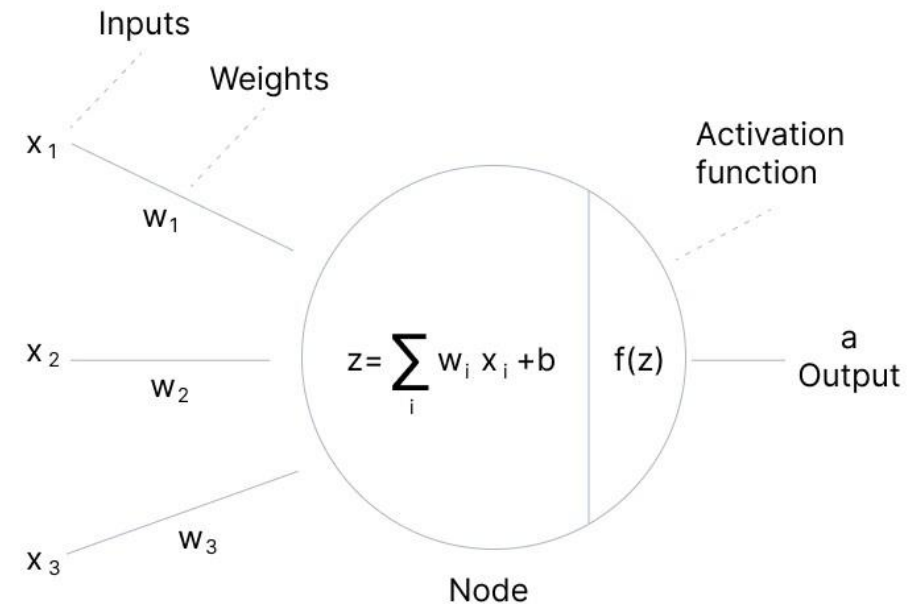
Introduction

What are activation functions?



Activation Function:

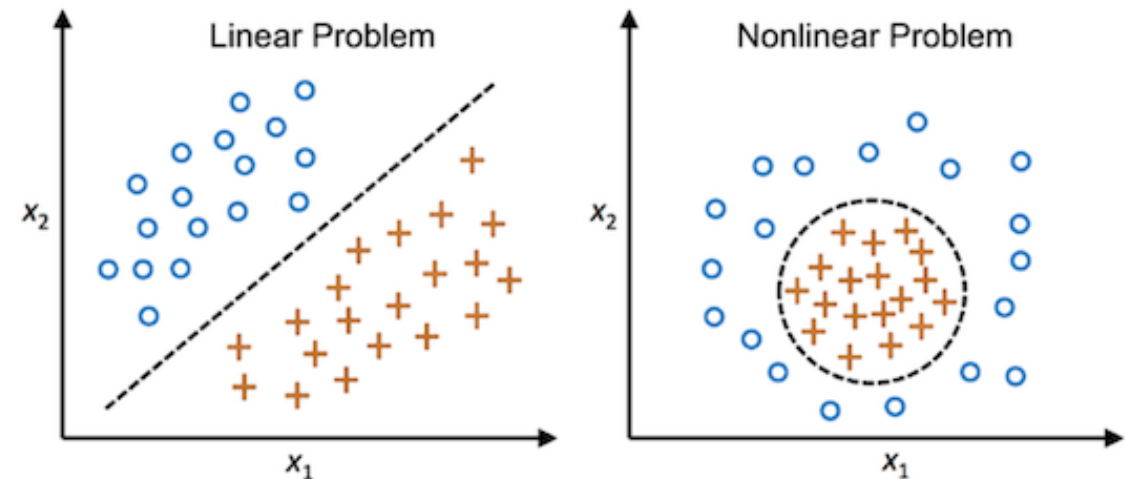
- is a function applied to the output of each neuron in a neural network layer
- decides whether a neuron should be activated by calculating the weighted sum of inputs and adding a bias term
- introduces non-linearity



Why activation functions are important?

Activation functions are crucial because they:

- Non-linearity
 - Modelling functions that are not linearly separable
 - Control of the flow of gradients during backpropagation, which affects convergence
 - learning more complex and abstract patterns from data.



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

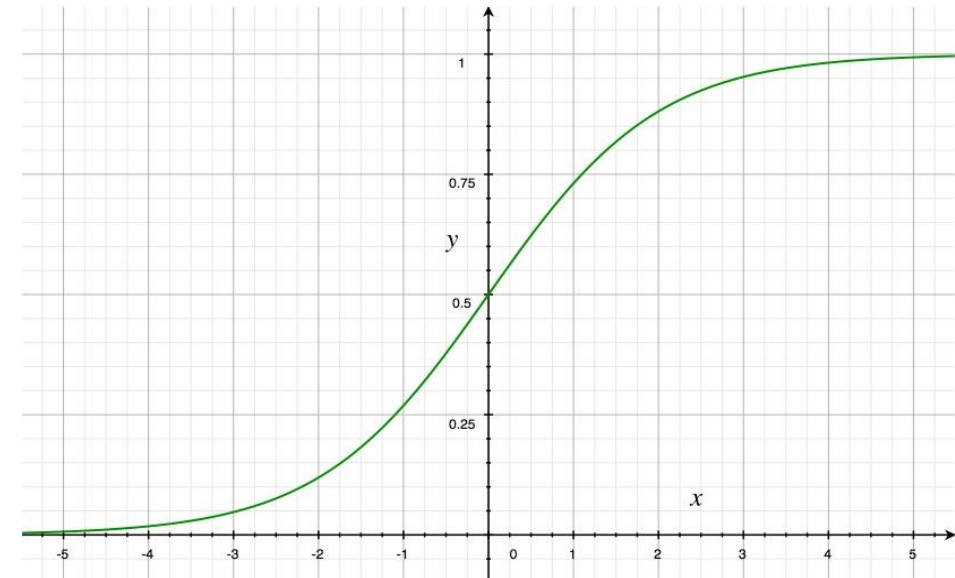
$$\sigma'(x) = \sigma(x) * (1 - \sigma(x))$$

Domain and Range:

- **Domain:** $\mathbb{R}$
- **Range:** $(0, 1)$

Derivative Properties:

- The maximum value of $\sigma'(x)$ occurs at $x = 0$: $\sigma'(x) = 0.25$
- For $|x| > 3$, the derivative becomes very small



Advantages

- Smooth and Continuous
- Bounded Output
- Simple Derivative

Disadvantages

- Vanishing Gradient Problem
- Not Zero-Centred
- Computational Expense
- Saturation Issues

Optimal Use Cases

- Binary Classification Output
- Gate Mechanisms in LSTM/GRU
- Attention Mechanisms.
- Probability Estimation

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

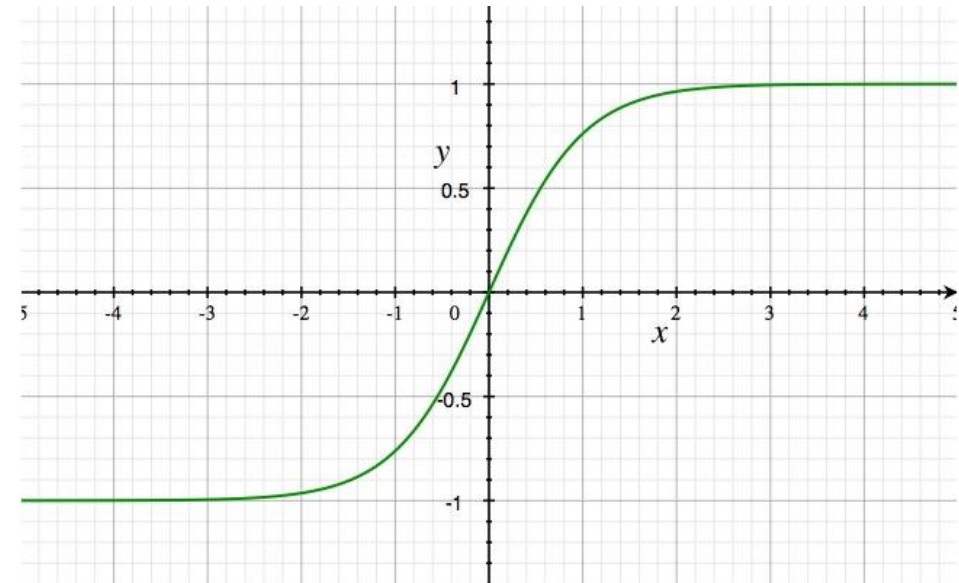
$$\tanh'(x) = 1 - \tanh^2(x)$$

Derivative Properties:

- The maximum occurs at $x = 0$: $\tanh'(0) = 1$
- For large $|x|$, the derivative approaches zero

Domain and Range:

- **Domain:** $\mathbb{R}$
- **Range:** $(-1, 1)$



Advantages

- Zero-Centered Output
- Stronger Gradients
- Steeper Activation Curve
- Symmetric Activation

Disadvantages

- Persistent Vanishing Gradient
- Computational Complexity
- Saturation at Extremes
- Limited Applicability in Very Deep Networks

Optimal Use Cases

- Hidden Layers in Shallow to Medium Networks
- Recurrent Neural Networks (RNNs)
- LSTM Cell State Processing
- Data with Symmetric Properties

$$f(x) = \max(0, x) = \begin{cases} x, & \text{if } x > 0, \\ 0, & \text{if } x \leq 0 \end{cases}$$

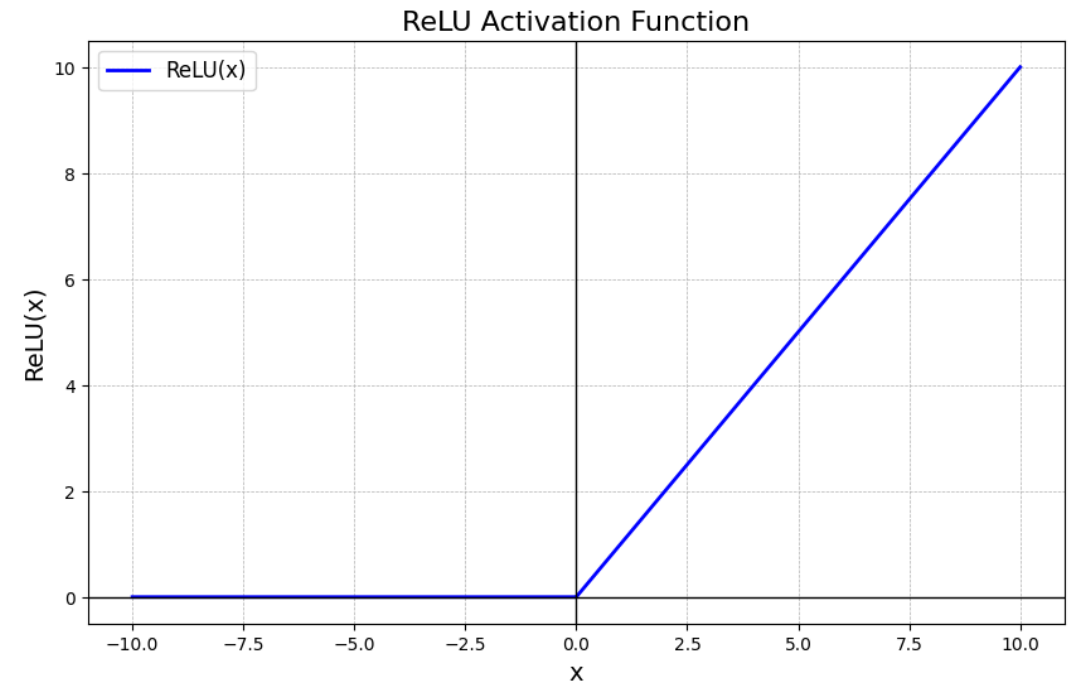
$$f'(x) = \begin{cases} 1, & \text{if } x > 0, \\ 0, & \text{if } x \leq 0 \end{cases}$$

Derivative Properties

- Constant Gradient
- Zero Gradient
- Binary Nature
- No Gradient Vanishing

Output Range

- **Range:** $[0, +\infty)$
- **Domain:** $(-\infty, +\infty)$



Advantages:

- Computational Efficiency
- Sparse Activation
- No Vanishing Gradient
- Linear Behavior

Disadvantages:

- Dying ReLU
- Dead Neuron Problem
- Not Zero-Centred
- Unbounded
- Zero gradient for $x < 0$

Typical Use Cases:

- Hidden Layers
- Convolutional Neural Networks
- Deep Learning
- Computer Vision

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

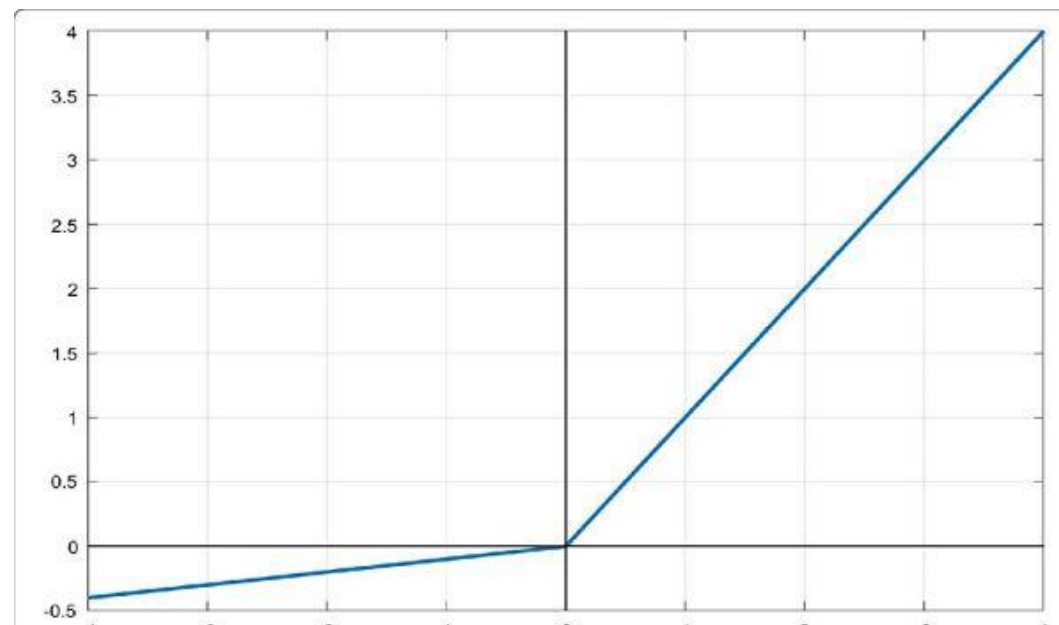
$$f'(x) = \begin{cases} 1, & \text{if } x > 0, \\ \alpha, & \text{if } x \leq 0 \end{cases}$$

Derivative Properties

- Consistent Positive Gradient
- Asymmetric Gradients
- Continuous Gradient Flow
- Parameter Dependent
- Bounded Below

Output Range

- **Range:** $(-\infty, +\infty)$
- **Domain:** $(-\infty, +\infty)$



Advantages:

- No Dead Neurons
- Computational Efficiency
- Maintains ReLU Benefits

Disadvantages:

- Hyperparameter Tuning
- Not Zero-Centred
- Inconsistent Performance

Typical Use Cases:

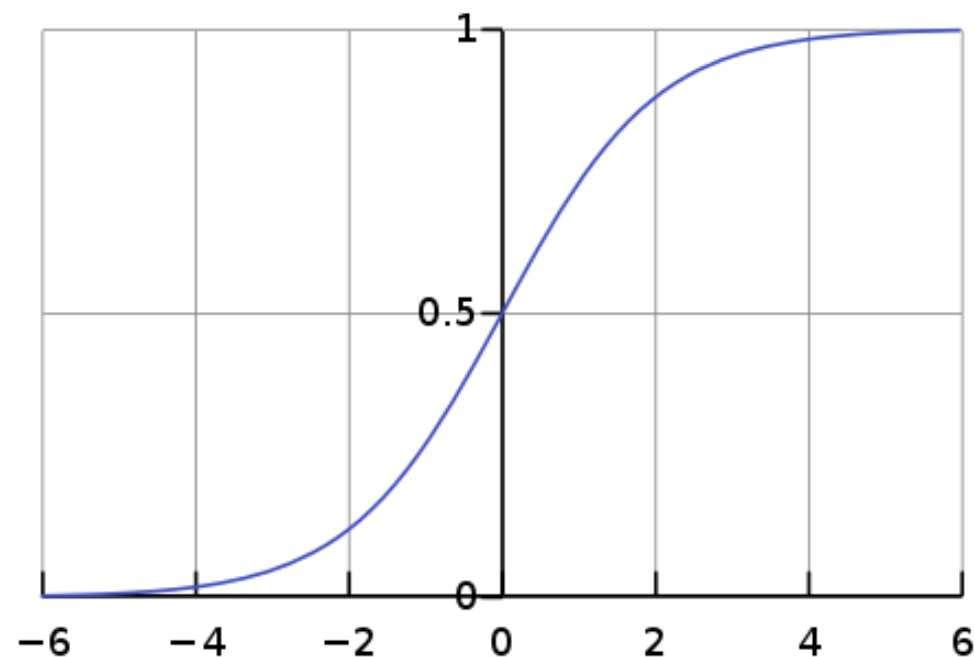
- Deep Networks
- Experimental Architectures
- Transfer Learning
- Research Applications

$$f(z_{b_j}) = \frac{e^{z_c}}{\sum_{c'}^C e^{z_{c'}}}$$

$$\frac{\partial f(z_i)}{\partial z_j} = \begin{cases} f_i(1 - f_i), & \text{if } i = j \\ -f_i f_j, & \text{if } i \neq j \end{cases}$$

Output Range

- **Range:** (0,1)
- **Domain:** $(-\infty, +\infty)$



Advantages:

- Probabilistic interpretation
- Works with cross-entropy loss
- Differentiable everywhere
- Handles multi-class naturally

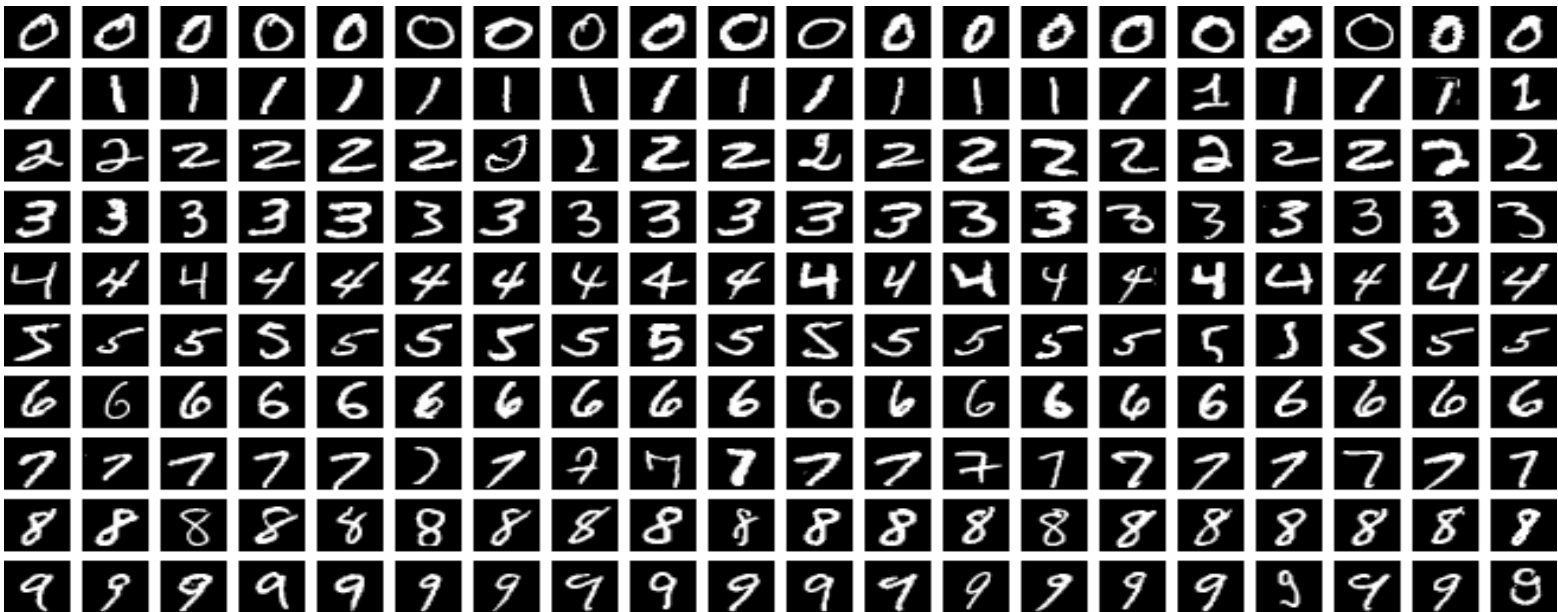
Disadvantages:

- Computational Complexity
- Overflow Risk
- Saturation
- Not Suitable for Hidden Layers

Typical Use Cases:

- Multi-class Classification
- Attention Mechanisms
- Output Layer
- Temperature Scaling

We trained a fully connected neural network on the MNIST handwritten digit dataset using different activation functions to study their effect on learning and generalization.



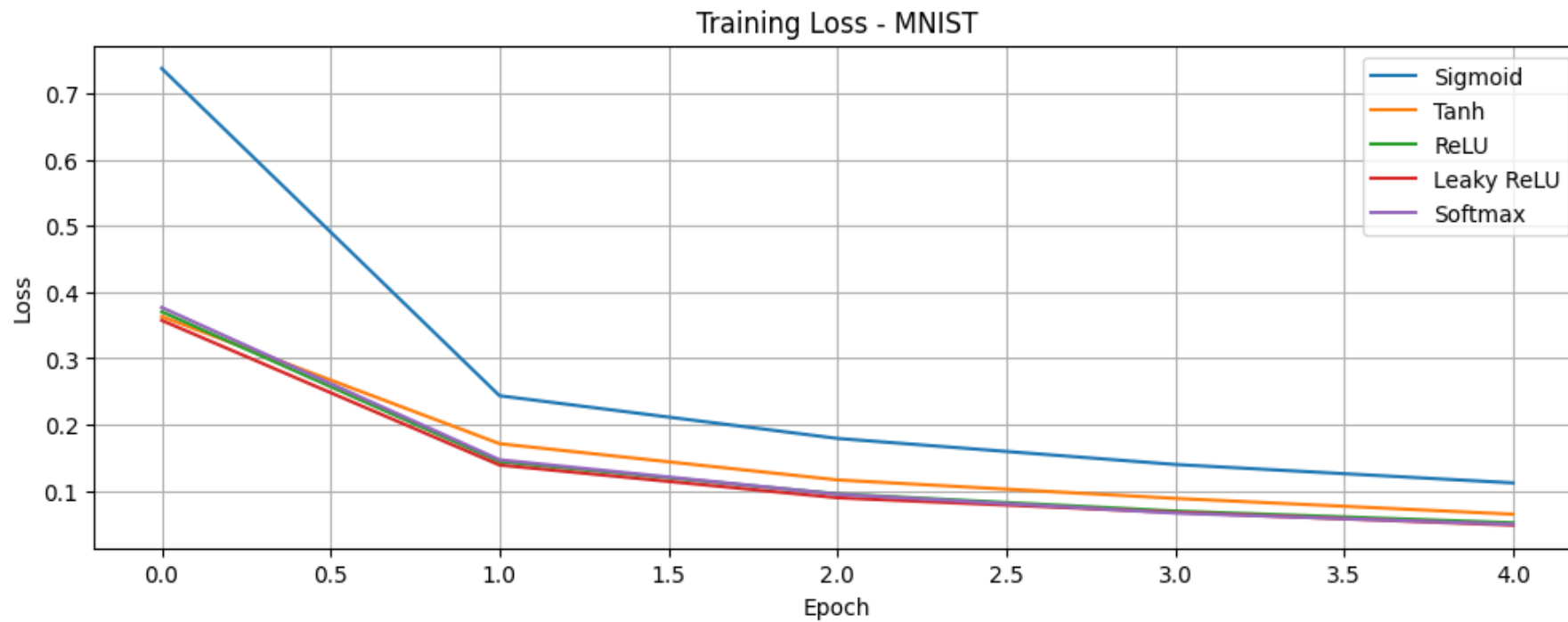
- The dataset consists of 28×28 grayscale images of digits (0–9), flattened into 784-length vectors.
- **Model Architecture:**
 - Input layer: 784 units (flattened 28×28 image)
 - Hidden Layer 1: 256 units + Activation
 - Hidden Layer 2: 128 units + Activation
 - Output Layer: 10 units (one per digit class)

Data Handling

- We used the MNIST CSV dataset from Kaggle.
- Data was normalized to range $[0, 1]$, then split into 80% training and 20% testing using sklearn.
- PyTorch tensors and DataLoaders were used for efficient batching and training.

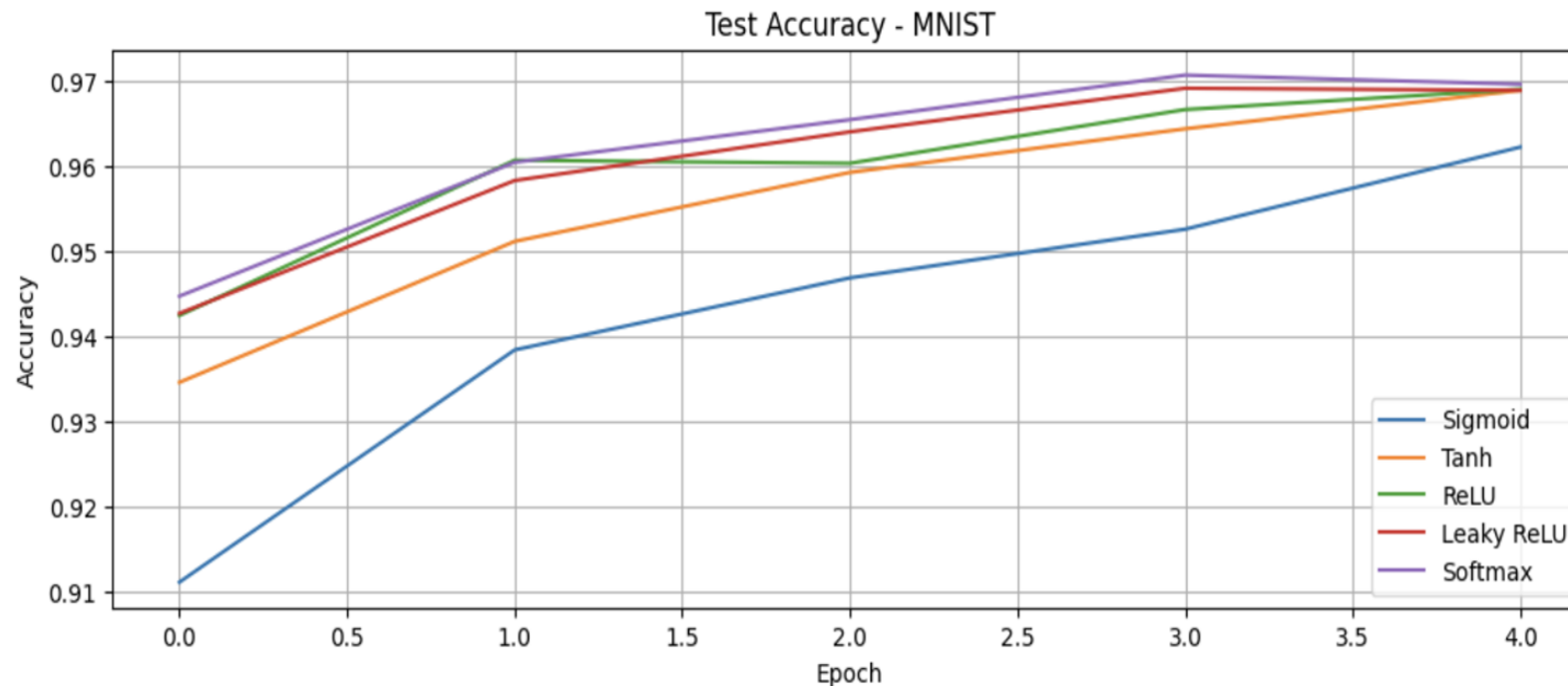
Training Strategy

- We trained each model for 5 epochs using Adam optimizer and CrossEntropyLoss.
- CrossEntropyLoss handles classification and internally applies Softmax, so we didn't apply it manually except during test time for the Softmax case.



Testing Accuracy

- Test accuracy tells us how well the model generalizes to unseen data.
- It's the key indicator for real-world performance.
- High training accuracy doesn't mean much if test accuracy is low (overfitting).



- We focused on activation function impact, not architecture. CNNs would add complexity.
- Add more epochs. This was to observe early-stage learning behaviour and keep experiments fast and comparable.
- Try deeper networks, CNNs, or batch normalization with different activations.

Takeaways

- Activation functions introduce non-linearity
- Choice of activation function greatly affects training speed and accuracy
- There is no “best” activation function-choice depends on the task, data, type of layer

Thank you for your attention!

Any Questions? 😊