Activation Functions in Deep Learning

By

Mausam Chouksey

September 26, 2024

Introduction

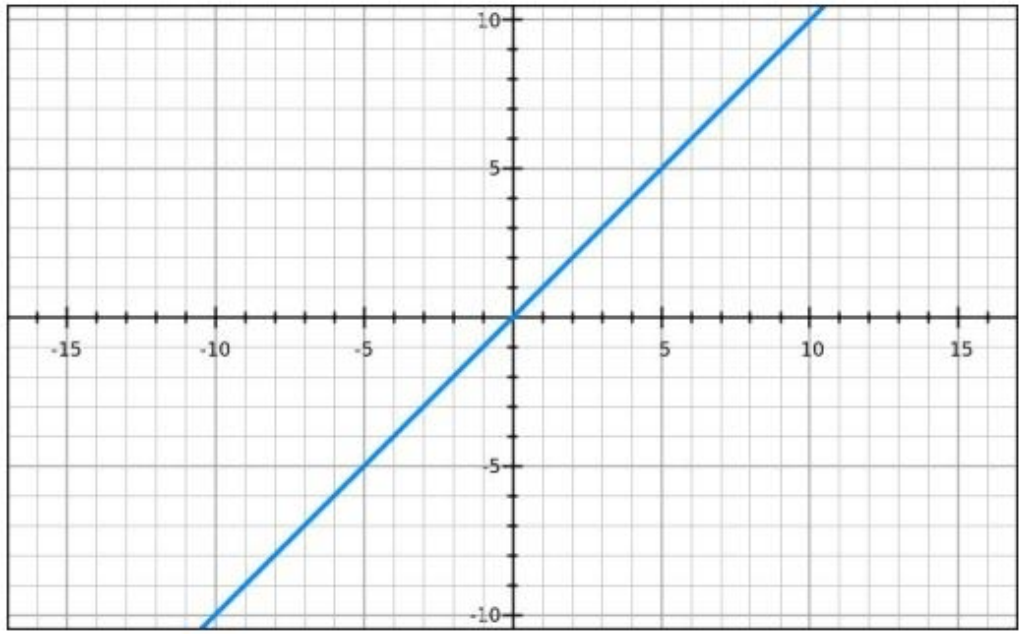
- We use activation functions to propagate the output of one layer's nodes forward to the next layer .
- Activation functions are a scalar-to-scalar function, yielding the neuron's activation.
- We use activation functions for hidden neurons in a neural network to introduce nonlinearity into the network's modeling capabilities.
- Many activation functions belong to a logistic class of transforms that (when graphed) resemble an S. This class of function is called sigmoidal. The sigmoid family of functions contains several variations, one of which is known as the Sigmoid function.

Linear Activation Function (Identity)

Formula:

$$f(x) = x$$

Range: $(-\infty, \infty)$



Cont'd...

Use Cases:

- Used in the output layer for regression tasks (predicting continuous values).

Advantages:

- Simple and computationally efficient.
- Useful for regression tasks where the output needs to be a continuous value.

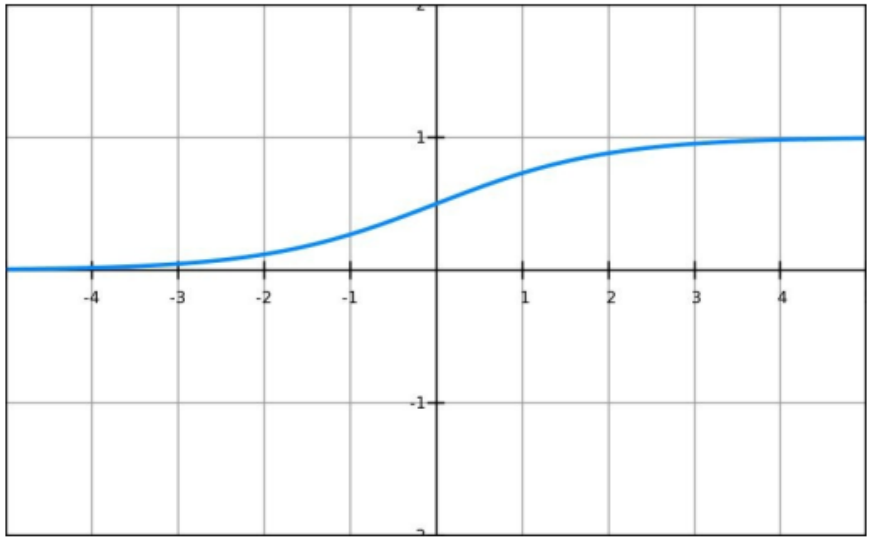
Limitations:

- No non-linearity, thus it can't capture complex relationships in the data.
- Can't be used in hidden layers, as the network becomes equivalent to a linear model regardless of depth.
- Unbounded output can lead to instability with large inputs.

Sigmoid Activation Function

Formula:

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



Cont'd…

Range: (0, 1)
Use Cases:

- Binary classification (logistic regression)
- Output layer for binary classification tasks

Advantages:

- Provides probabilistic interpretation (outputs between 0 and 1)
- Smooth gradient for gradient descent

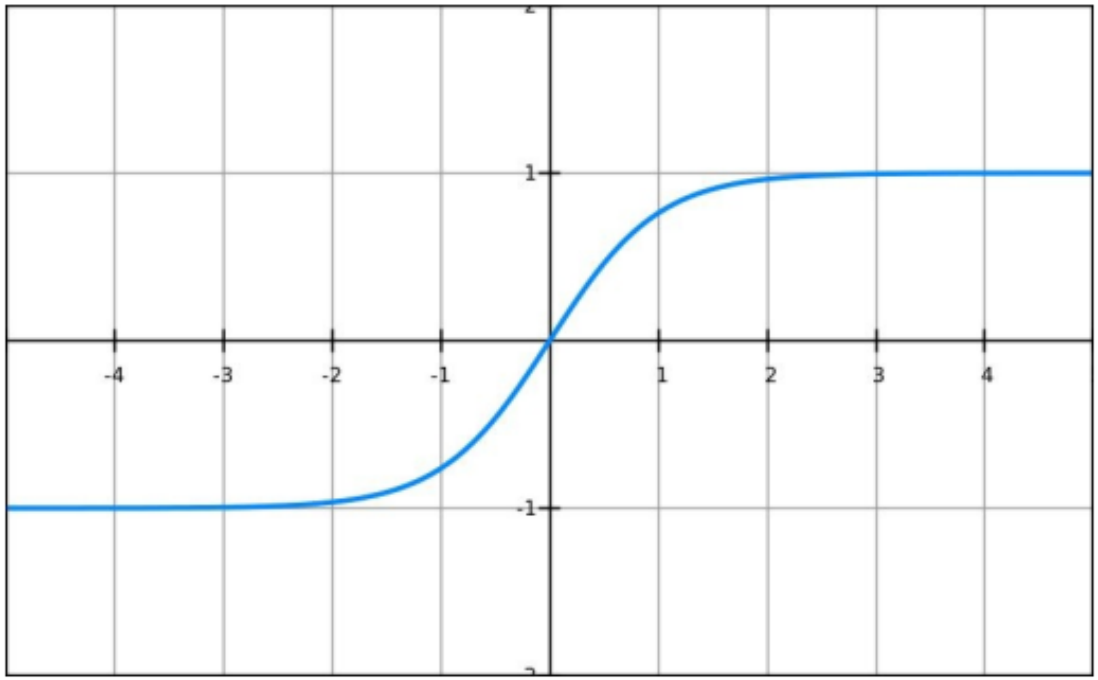
Limitations:

- **Vanishing gradient problem:** For very large or very small inputs, the gradient becomes almost zero, slowing training
- Outputs are not zero-centered, which may result in slow convergence

Tanh Activation Function

Formula:

$$\tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$



Cont'd…

Range: (-1, 1)
Use Cases:

- Hidden layers of neural networks

Advantages:

- Outputs are zero-centered (helps in faster optimization)
- Smooth gradient like Sigmoid

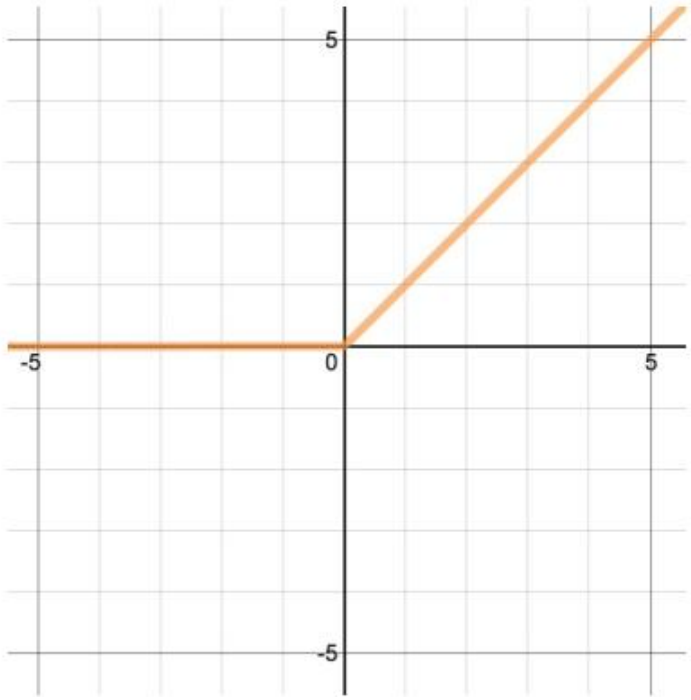
Limitations:

- Suffers from the vanishing gradient problem for large input values

ReLU Activation Function

Formula:

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



Cont'd…

Range: [0, ∞)

Use Cases:

- Most commonly used in hidden layers of deep neural networks (e.g., CNNs)

Advantages:

- Simple and computationally efficient
- Solves the vanishing gradient problem for positive values

Limitations:

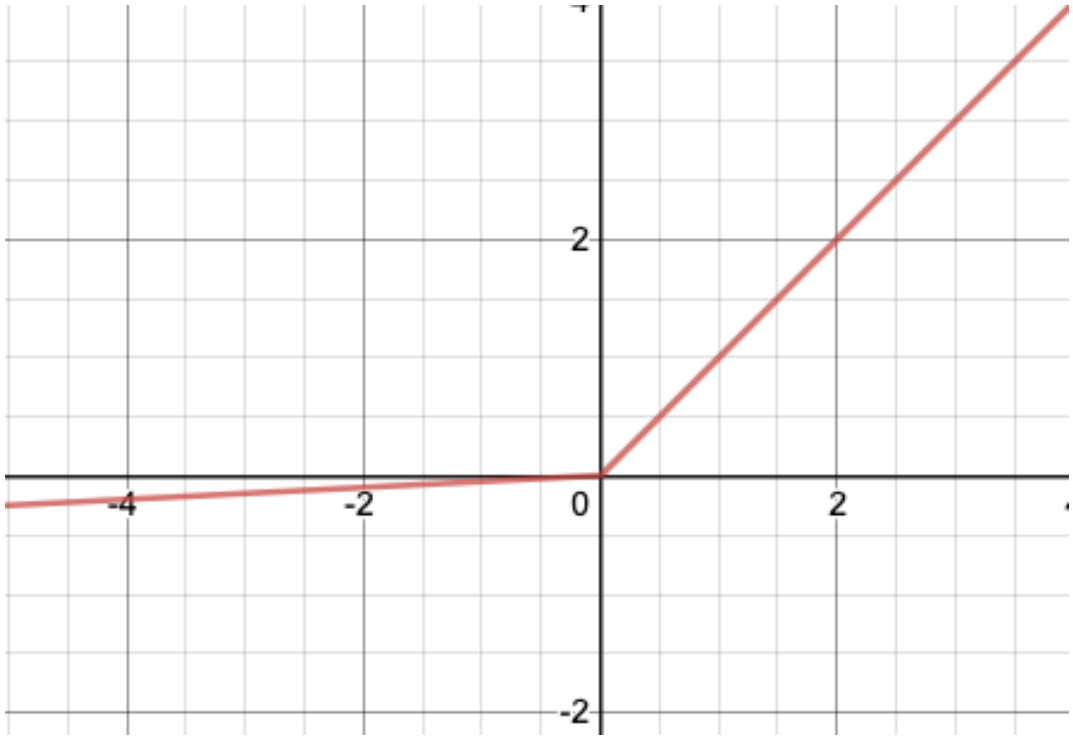
- **Dying ReLU problem:** Neurons can stop updating for negative inputs
- Unbounded outputs, leading to potential exploding gradients

Leaky ReLU Activation Function

Formula:

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

where a is a small constant, typically 0.01.



Cont'd...

Range: $(-\infty, \infty)$
Use Cases:

- Deep networks where the dying ReLU problem is observed

Advantages:

- Solves the dying ReLU problem by allowing small gradients when $x < 0$
- Computationally efficient

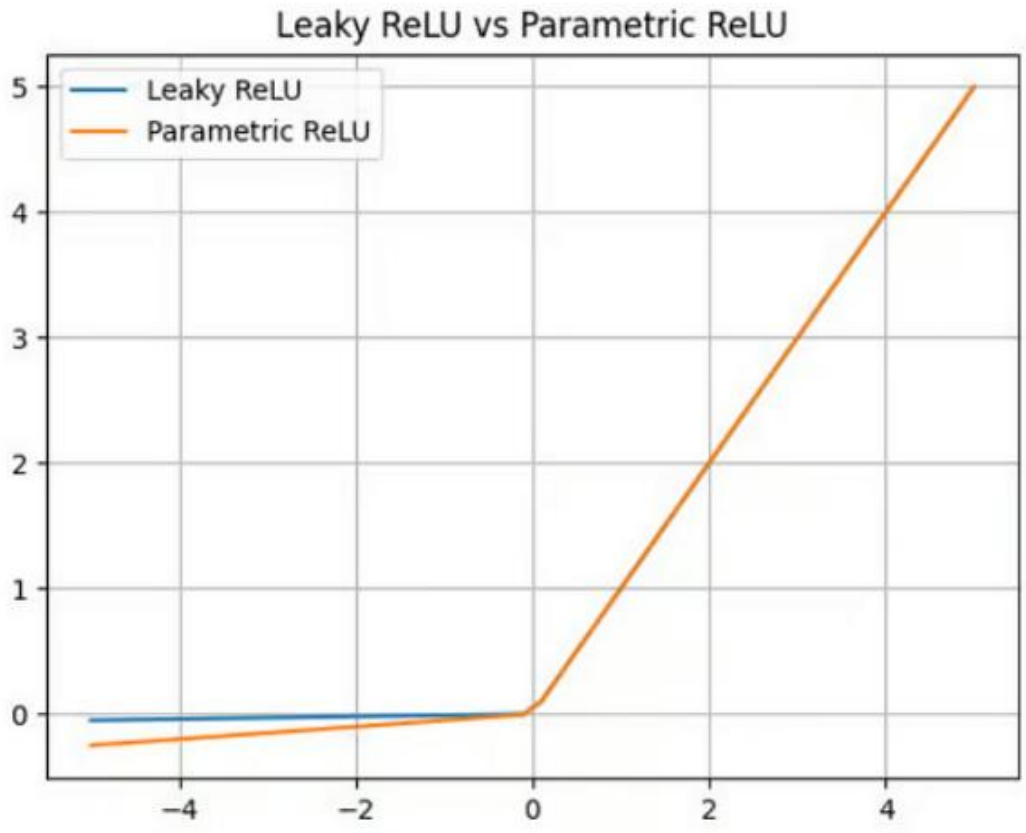
Limitations:

- Choosing a requires tuning
- Unbounded output may cause exploding gradients

Parametric ReLU (PReLU)

Formula:

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



Cont'd...

where a is a learnable parameter.
Range: $(-\infty, \infty)$
Use Cases:

- Deep networks, especially CNNs

Advantages:

- a is learned during training, potentially improving performance
- Solves the dying ReLU problem

Limitations:

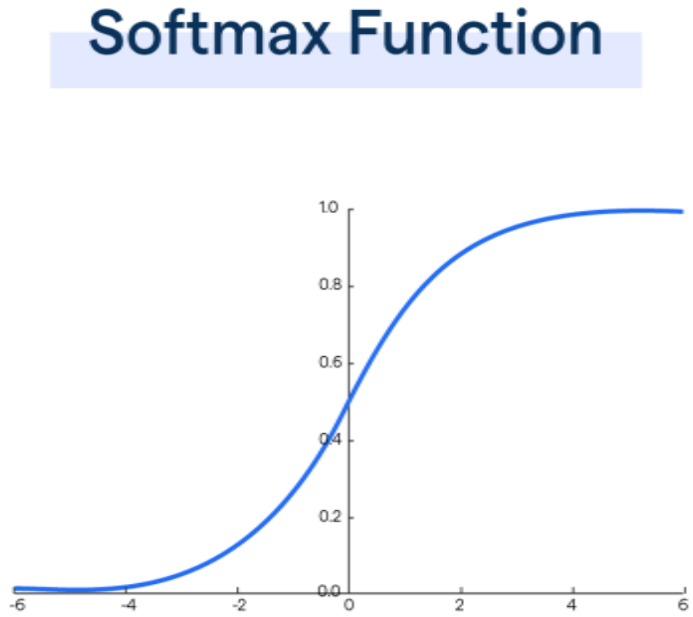
- Adds more parameters, increasing risk of overfitting
- Unbounded output can still cause exploding gradients

Softmax Activation Function

Formula:

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

Range: (0, 1)



Cont'd...

Range: (0, 1)
Use Cases:

- Output layer for multi-class classification tasks

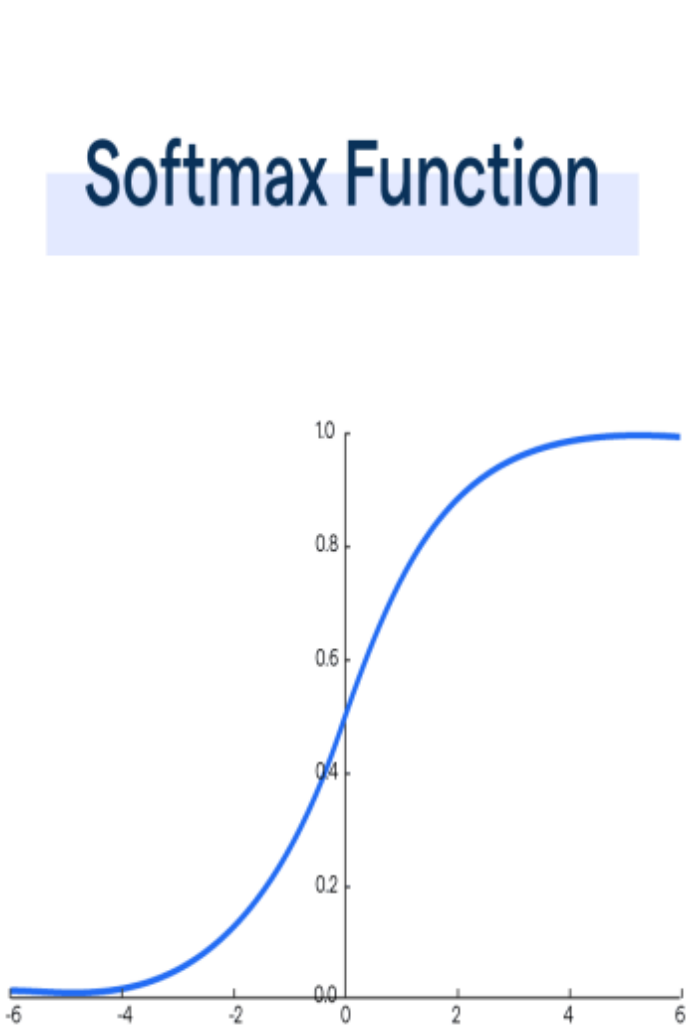
Advantages:

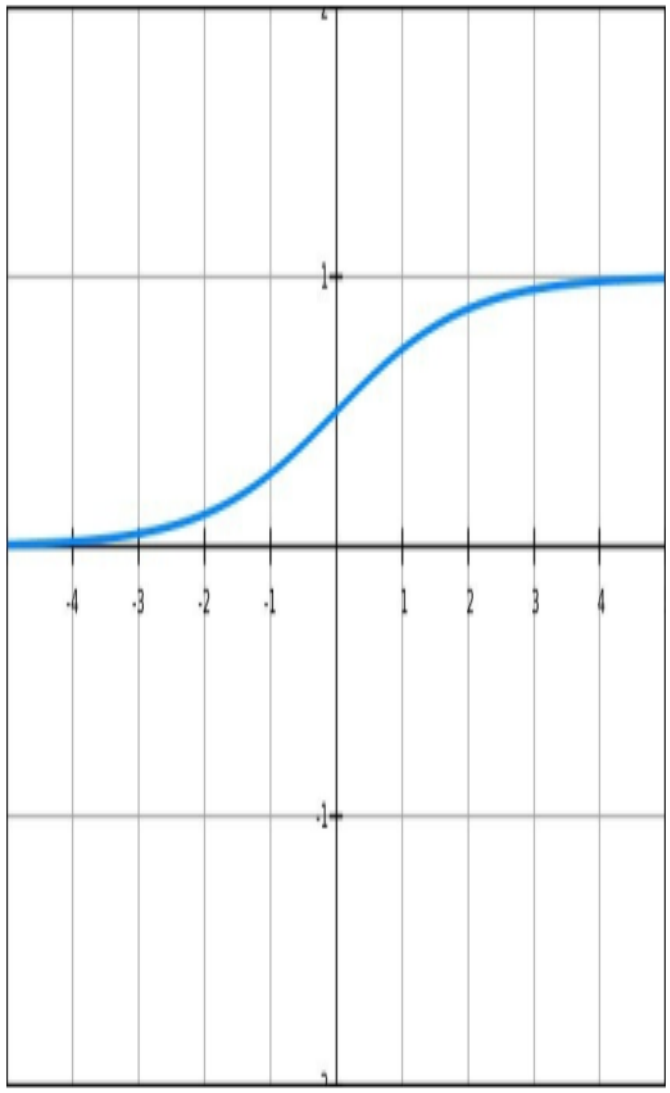
- Outputs can be interpreted as probabilities
- Well-suited for multi-class classification

Limitations:

- Computationally expensive for large output spaces
- Sensitive to large input values, leading to numerical instability

Softmax vs Sigmoid



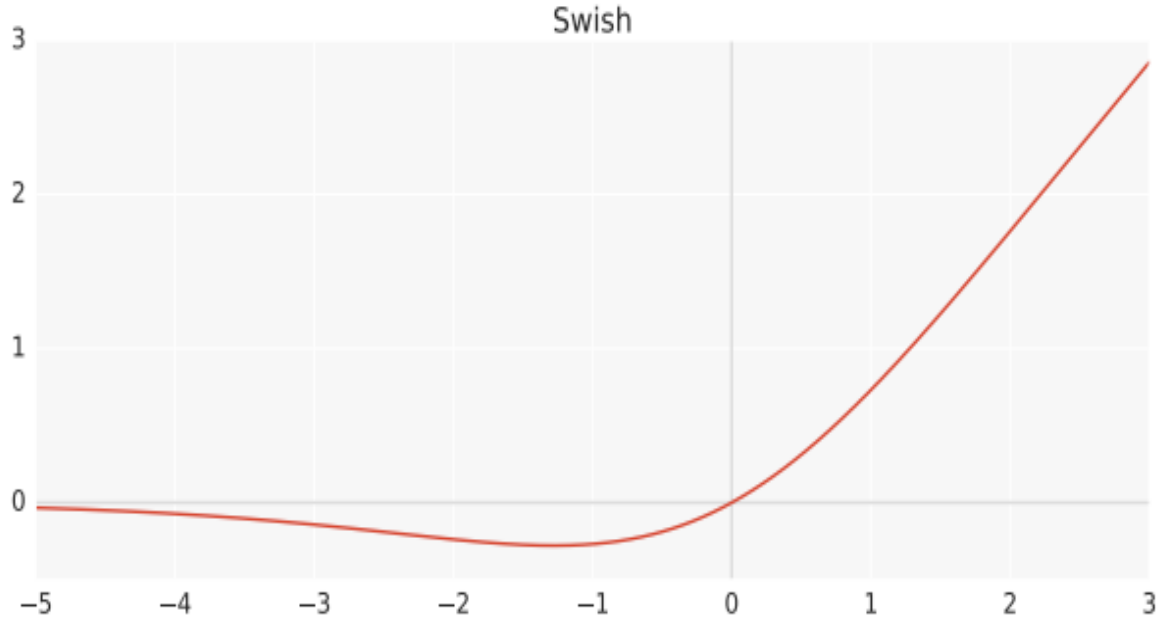


Swish Activation Function

Formula:

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

Range: $(-\infty, \infty)$



Cont'd...

Range: $(-\infty, \infty)$
Use Cases:

- Deep networks, often outperforms ReLU in deeper models

Advantages:

- Non-monotonic, allowing better information flow
- Smooth gradient, helping avoid vanishing gradient issues

Limitations:

- More computationally expensive than ReLU

Hidden layer activation functions

Commonly used functions include:

- Sigmoid
- Tanh
- Hard tanh
- Rectified linear unit (ReLU) (and its variants)

A more continuous distribution of input data is generally best modeled with a ReLU activation function.

Output layer activation functions

Output layer for regression

This design decision is motivated by what type of answer we expect our model to output. If we want to output a single real-valued number from our model, we'll want to use a linear activation function.

Output layer for binary classification

In this case, we'd use a sigmoid output layer with a single neuron to give us a real value in the range of 0.0 to 1.0 (excluding those values) for the single class. This real-valued output is typically interpreted as a probability distribution.

Output layer for multiclass classification

If we have a multiclass modeling problem yet we only care about the best score across these classes, we'd use a softmax output layer with an arg-max() function to get the highest score of all the classes. The softmax output layer gives us a probability distribution over all the classes.