What is an Activation Function?

An activation function is a mathematical function applied to the output of a neuron. It introduces non-linearity into the model, allowing the network to learn and represent complex patterns in the data. Without this non-linearity feature, a neural network would behave like a linear regression model, no matter how many layers it has.

The activation function decides whether a neuron should be activated by calculating the weighted sum of inputs and adding a bias term. This helps the model make complex decisions and predictions by introducing non-linearities to the output of each neuron.

Why is Non-Linearity Important in Neural Networks?

Neural networks consist of neurons that operate using **weights**, **biases**, and **activation functions**.

In the learning process, these weights and biases are updated based on the error produced at the output—a process known as **backpropagation**. Activation functions enable backpropagation by providing gradients that are essential for updating the weights and biases. Without non-linearity, even deep networks would be limited to solving only simple, linearly separable problems. Activation functions empower neural networks to model highly complex data distributions and solve advanced deep learning tasks. Adding non-linear activation functions introduce flexibility and enable the network to learn more complex and abstract patterns from data.

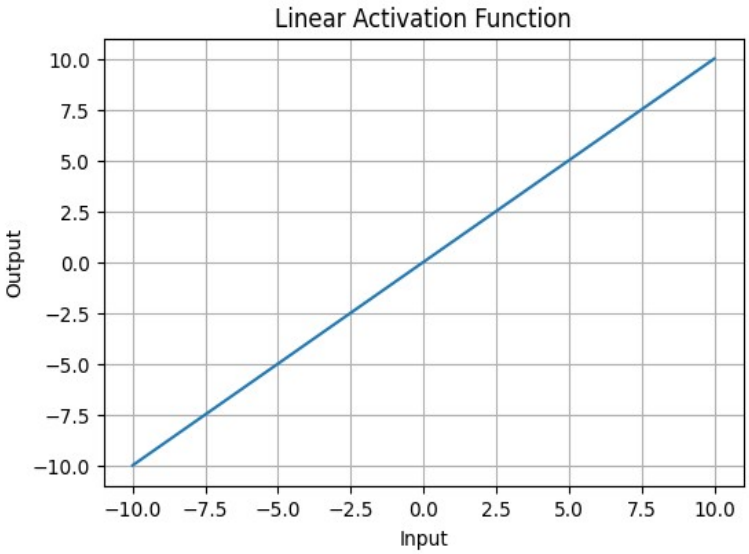
Types of Activation Functions in Deep Learning

1. Linear Activation Function

Linear Activation Function resembles straight line define by y=x. No matter how many layers the neural network contains, if they all use linear activation functions, the output is a linear combination of the input.

- The range of the output spans from (−∞ to +∞).
- **Linear activation function** is used at just one place i.e. output layer.
- Using linear activation across all layers makes the network's ability to learn complex patterns limited.

Linear activation functions are useful for specific tasks but must be combined with non-linear functions to enhance the neural network's learning and predictive capabilities.



Linear Activation Function or Identity Function returns the input as the output

2. Non-Linear Activation Functions

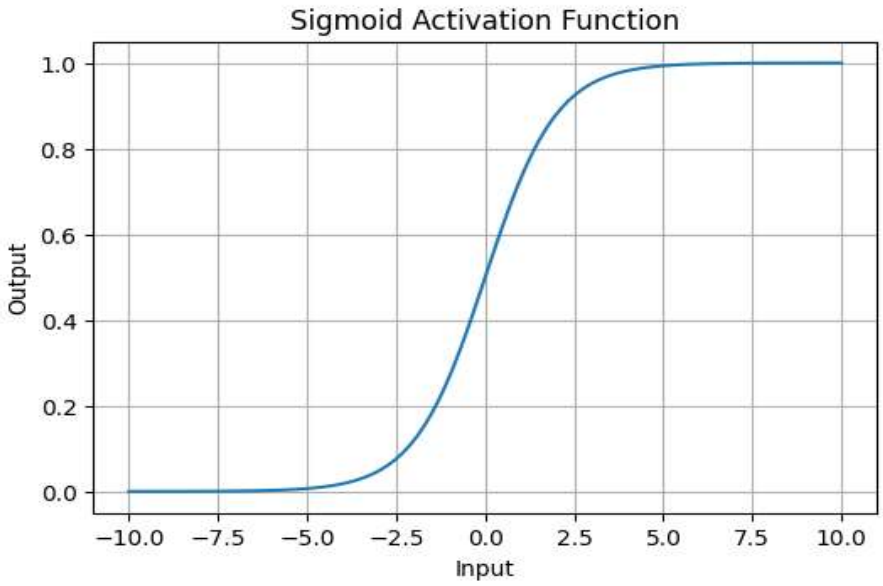
1. Sigmoid Function

Sigmoid Activation Function is characterized by 'S' shape.

It is mathematically defined as $Y = \frac{1}{(1 + e^{-x})}$

This formula ensures a smooth and continuous output that is essential for gradient-based optimization methods.

- It allows neural networks to handle and model complex patterns that linear equations cannot.
- The output ranges between 0 and 1, hence useful for binary classification.



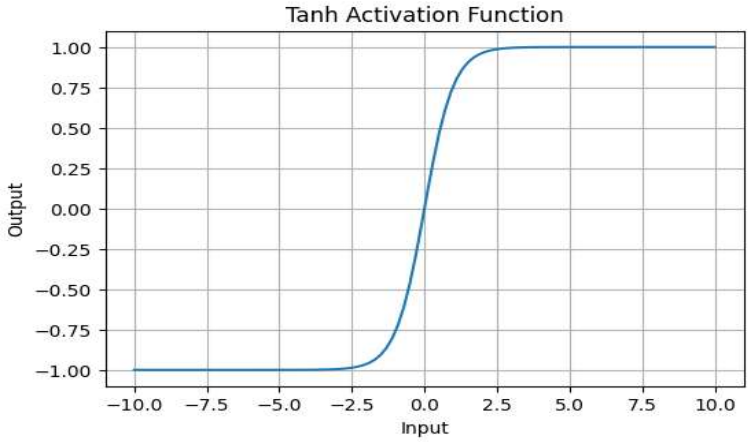
Sigmoid or Logistic Activation Function Graph

2. Tanh Activation Function

Tanh function or hyperbolic tangent function, is a shifted version of the sigmoid, allowing it to stretch across the y-axis. It is defined as:

$$y = \frac{1 - e^{-net}}{1 + e^{-net}}$$

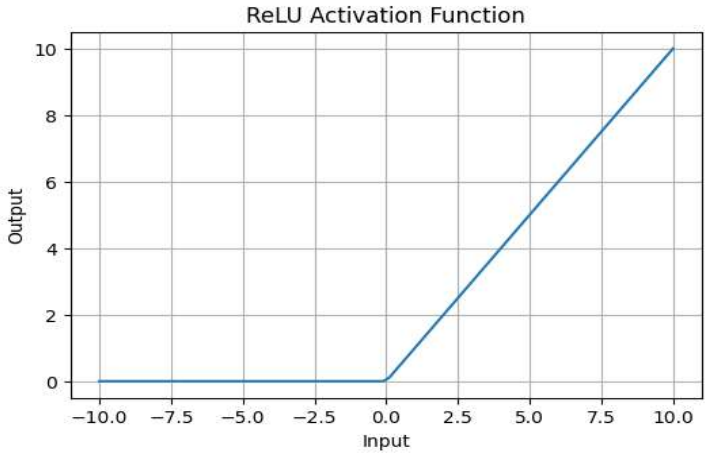
- **Value Range**: Outputs values from -1 to +1.
- **Non-linear**: Enables modeling of complex data patterns.
- **Use in Hidden Layers**: Commonly used in hidden layers due to its zero-centered output, facilitating easier learning for subsequent layers.



3. ReLU (Rectified Linear Unit) Function

ReLU activation is defined by A(x) = max(0,x) this means that if the input x is positive, ReLU returns x, if the input is negative, it returns 0.

- **Value Range**: [0,∞), meaning the function only outputs non-negative values.
- **Nature**: It is a **non-linear** activation function, allowing neural networks to learn complex patterns and making backpropagation more efficient.
- **Advantage over other Activation:** ReLU is less computationally expensive than tanh and sigmoid because it involves simpler mathematical operations.



3. Exponential Linear Units

1. Softmax Function

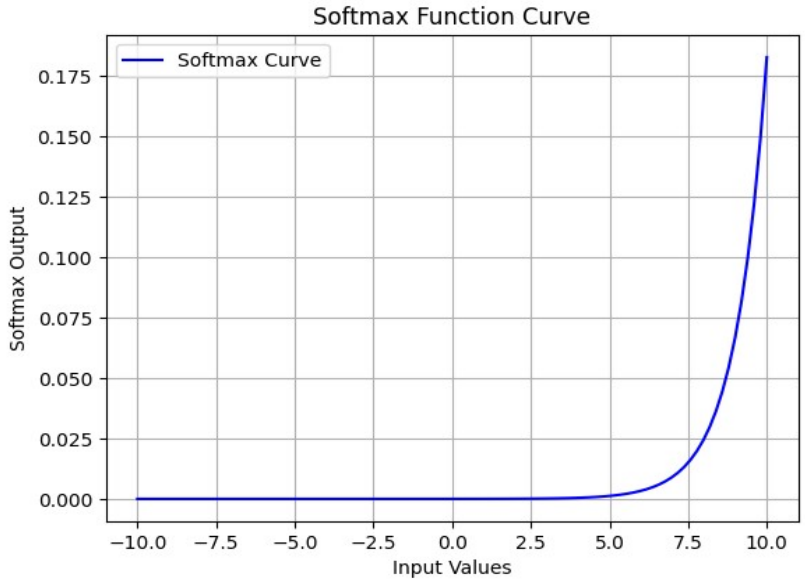
Softmax function is designed to handle multi-class classification problems. It transforms raw output scores from a neural network into probabilities. It works by squashing the output values of each class into the range of 0 to 1, while ensuring that the sum of all probabilities equals 1.

$$S(y)_i = \frac{\exp(y_i)}{\sum_{j=1}^{n} \exp(y_j)}$$

where,

y	is an input vector to a softmax function, S. It consist of n elements for n classes (possible outcomes)
y_i	the i-th element of the input vector. It can take any value between -inf and +inf
$exp(y_i)$	standard exponential function applied on y_i. The result is a small value (close to 0 but never 0) if $y_i < 0$ and a large value if y_i is large. eg • $\exp(55) = 7.69e{+}23$ (A very large value) • $\exp(-55) = 1.30e$-24 (A very small value close to 0) **Note**: $\exp(*)$ is just $\mathbf{e}^*$ where $\mathbf{e} = 2.718$, the Euler's number.
$\sum_{j=1}^{n} \exp(y_j)$	A normalization term. It ensures that the values of output vector $S(y)_i$ sums to 1 for i-th class and each of them and each of them is in the range 0 and 1 which makes up a valid probability distribution.
n	Number of classes (possible outcomes)

- Softmax is a **non-linear** activation function.
- The Softmax function ensures that each class is assigned a probability, helping to identify which class the input belongs to.



Impact of Activation Functions on Model Performance

The choice of activation function has a direct impact on the performance of a neural network in several ways:

1. **Convergence Speed:** Functions like **ReLU** allow faster training by avoiding the vanishing gradient problem, while **Sigmoid** and **Tanh** can slow down convergence in deep networks.
2. **Gradient Flow:** Activation functions like **ReLU** ensure better gradient flow, helping deeper layers learn effectively. In contrast, **Sigmoid** can lead to small gradients, hindering learning in deep layers.
3. **Model Complexity:** Activation functions like **Softmax** allow the model to handle complex multi-class problems, whereas simpler functions like **ReLU** or **Leaky ReLU** are used for basic layers.