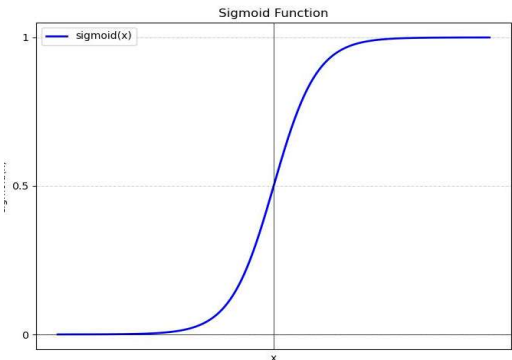
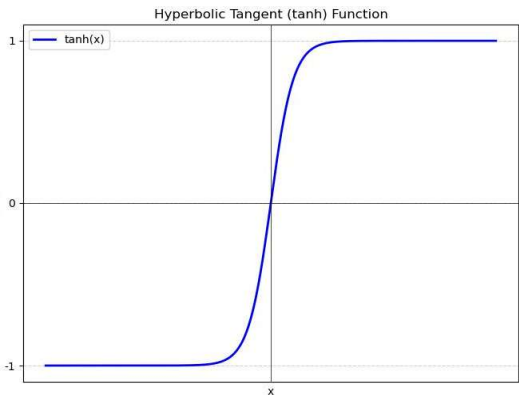


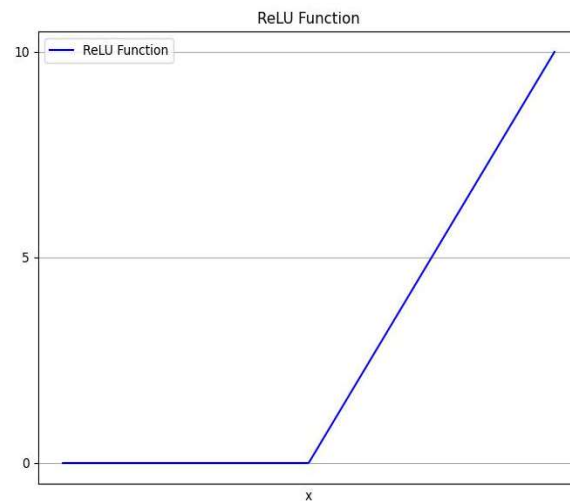
Fundamentals of Deep Learning

Activation Functions

Function	Graph	Characteristics
<p>Sigmoid Function</p> $\sigma(z) = \frac{1}{1+e^{-z}}$	 <p>The graph shows the Sigmoid Function, which is an S-shaped curve. The x-axis is labeled 'x' and the y-axis is labeled 'sigmoid(x)'. The curve starts near 0 for negative x, passes through (0, 0.5), and approaches 1 for positive x. The y-axis has tick marks at 0, 0.5, and 1.</p>	<ol style="list-style-type: none">1. Values between 0 and 12. Suitable for outputs with binary classification or probabilities3. Suffers from vanishing gradient problem
<p>Hyperbolic Tangent Function</p> $\tanh(z) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	 <p>The graph shows the Hyperbolic Tangent (tanh) Function, which is an S-shaped curve centered at the origin. The x-axis is labeled 'x' and the y-axis is labeled 'tanh(x)'. The curve ranges from -1 to 1, passing through (0, 0). The y-axis has tick marks at -1, 0, and 1.</p>	<ol style="list-style-type: none">1. Values between -1 and +12. Convergence is a bit faster3. Suffers from vanishing gradient problem4. Smooth and differentiable

Rectified Linear Unit (ReLU)

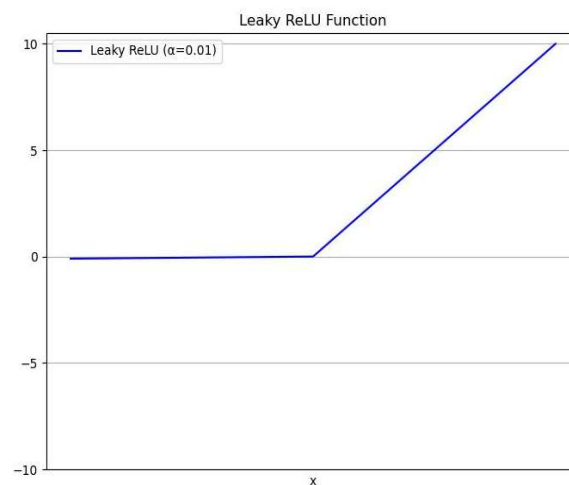
$$\phi(x) = \begin{cases} 0 & x \leq 0 \\ x & x > 0 \end{cases}$$



1. Values can range from 0 to ∞
2. Some nodes with little information may be zeroed out (Sparse activation)
3. Avoids vanishing gradient problem for positive inputs
4. Can suffer from dying ReLU problem due to high learning rates, biases etc.
5. Useful to capture large effects

Leaky Rectified Linear Unit (Leaky ReLU)

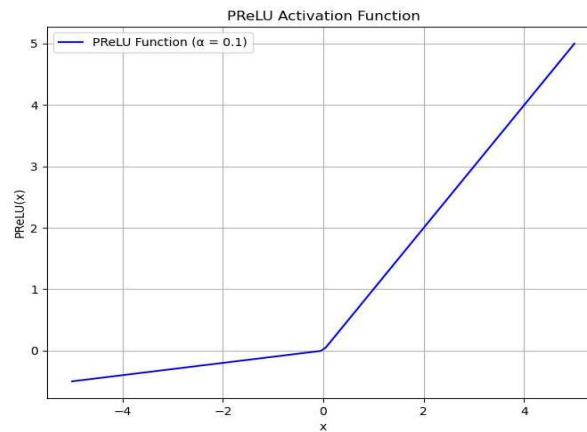
$$\phi(x) = \begin{cases} \alpha x & x \leq 0 \\ x & x > 0 \end{cases}$$



1. Similar to ReLU but allows negative outcomes
2. Values can range from $-\infty$ to $+\infty$
3. Outputs are scaled by a factor α (learning rate)
4. Fixed Dying ReLU problem
5. No nodes are zeroed out*

Parametric Rectified Linear Unit (PReLU)

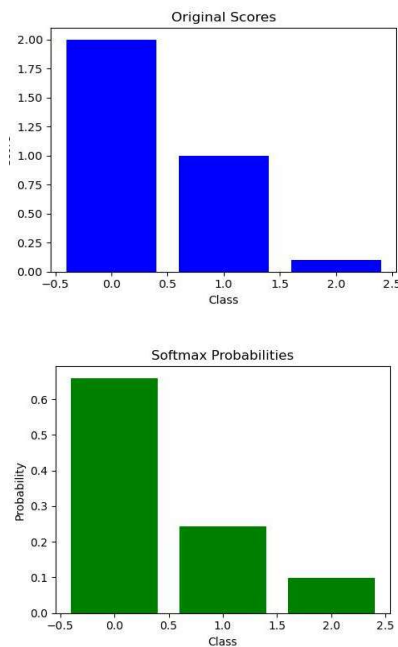
$$\text{PReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha \cdot x & \text{if } x \leq 0 \end{cases}$$



1. Extension of Leaky ReLU
2. Learns optimal slope of negative values in training using parameter α
3. Used in Convolutional Neural Networks

Softmax Function

$$s(x_i) = \frac{e^{z_i}}{\sum_{j=1}^n e^{z_j}}$$



1. Returns a vector of probabilities
2. Each element represents the probability of the corresponding class
3. Suitable for multi-class classification tasks, often in output layer
4. Outputs are scaled down
5. Smooth and differentiable