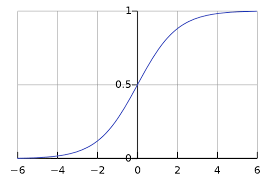
**1. Sigmoid (Logistic) Activation Function**





**Advantages:**

* **Smooth gradient**: Helps with gradient-based optimization methods.
* **Output range**: (0, 1), which can be interpreted as probabilities.

**Disadvantages:**

* **Vanishing gradient problem**: For very high or very low input values, the gradient becomes very small, which can slow down the training.
* **Outputs not zero-centered**: This can slow down convergence during gradient descent.

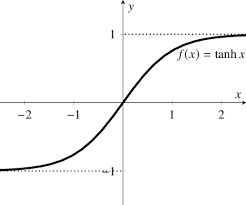
**Usage Recommendations:**

* **Sigmoid**: Historically used but less common now due to the vanishing gradient problem.

**2. Hyperbolic Tangent (tanh) Activation Function**

A black and white text

Description automatically generated



**Advantages:**

* **Zero-centered**: The outputs range from -1 to 1, which can help with convergence.
* **Smooth gradient**: Like the sigmoid, it provides a smooth gradient.

**Disadvantages:**

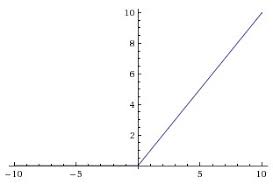
* **Vanishing gradient problem**: Similar to sigmoid, though less severe.

**Usage Recommendations:**

* **Tanh**: Historically used but less common now due to the vanishing gradient problem.

**3. Rectified Linear Unit (ReLU) Activation Function**





**Advantages:**

* **Computationally efficient**: Only requires a threshold at zero.
* **Sparse activation**: A portion of neurons are deactivated, promoting sparsity and efficient computation.
* **Alleviates vanishing gradient problem**: Increases the convergence rate of gradient descent.

**Disadvantages:**

* **Dying ReLU problem**: Neurons can sometimes get stuck during training and output zero for all inputs. This can be mitigated using variants like Leaky ReLU.

**Usage Recommendations:**

* **ReLU**: The default choice for most deep learning models due to its simplicity and effectiveness.

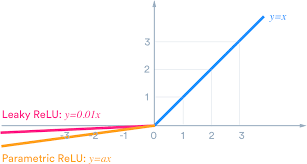
**4. Leaky ReLU Activation Function**

A group of math symbols

Description automatically generated

f(x)=max (αx, x)

Where α is a small constant, typically 0.01.



**Advantages:**

* **Solves dying ReLU problem**: Allows a small, non-zero gradient when the unit is not active.

**Disadvantages:**

* **Computationally slightly less efficient**: Due to the added multiplication.

**Usage Recommendations:**

* **Leaky ReLU:** Used when the dying ReLU problem is encountered.

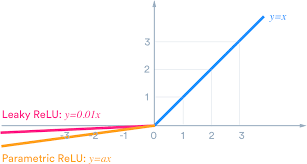
**5. Parametric ReLU (PReLU) Activation Function**

A group of math symbols

Description automatically generated

f(x)=max (αi x, x)

Where αi is a parameter learned during training.



**Advantages:**

* **Adaptive**: The parameter α\alphaα can be learned to best fit the data.

**Disadvantages:**

* **Increased computational cost**: Due to learning an additional parameter.

**Usage Recommendations:**

* **PReLU:** Used when the dying ReLU problem is encountered.

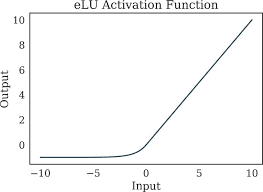
**6. Exponential Linear Unit (ELU)**

A close up of a number

Description automatically generated

f(x)=max (α(ex - 1), x)

where α is a positive constant that determines the value to which an ELU saturates for negative net inputs.



**Advantages:**

* **Improves learning characteristics**: Combines the benefits of ReLU and leaky ReLU.
* **Smooth gradient**: Reduces the vanishing gradient problem.

**Disadvantages:**

* **Computational cost**: More complex than ReLU.

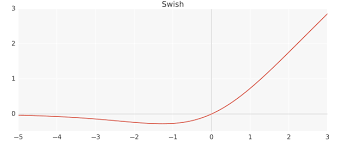
**Usage Recommendations:**

* **ELU**: Can be used when additional performance is needed, and computational resources allowed.

**7. Swish Activation Function**

A black text on a white background

Description automatically generated



**Advantages:**

* **Smooth and non-monotonic**: Often outperforms ReLU on deep networks.

**Disadvantages:**

* **Computationally more expensive**: Due to the combination of multiplication and sigmoid function.

**Usage Recommendations:**

* **Swish**: Can be used when additional performance is needed, and computational resources allowed.

**8. Usage Recommendations:**

* **Sigmoid and Tanh**: Historically used but less common now due to the vanishing gradient problem.
* **ReLU**: The default choice for most deep learning models due to its simplicity and effectiveness.
* **Leaky ReLU / PReLU**: Used when the dying ReLU problem is encountered.
* **ELU and Swish**: Can be used when additional performance is needed and computational resources allow.

