Activation Functions in Neural Networks

Activation functions determine how the weighted sum of inputs is transformed in a neuron. They introduce non-linearity, enabling the network to learn complex patterns. Below is a detailed explanation:

---

1. Linear Activation Function

Definition:

- Outputs the input as it is: $f(x) = x$

Usage:

- Rarely used in hidden layers due to its limitations.
- Used in the last layer for regression tasks.

Problems:

- **No non-linearity**: Cannot capture complex patterns.
- **Vanishing Gradient Problem**: Gradient remains constant, hindering weight updates.

Solution:

- Use non-linear functions in hidden layers.

Value Range:

- $(-\infty, +\infty)$

---

2. Non-Linear Activation Functions

These introduce non-linearity, enabling the network to learn complex relationships.

Types of Non-Linear Activation Functions:

a. Binary Step Function:

- **Definition**: Outputs 0 or 1 based on a threshold:

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

- **Usage:** Rarely used; mainly in binary classification (e.g., perceptrons).
- **Problems**:
  - Not differentiable.
  - Cannot propagate gradients.
- **Solution**: Use differentiable functions like sigmoid.
- **Value Range**: $\{0, 1\}$

---

b. Sigmoid:

- **Definition**:

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

- **Usage**: Often used in the last layer for binary classification.
- **Problems**:
  - Vanishing gradient for large positive/negative inputs.
  - Output not centered around zero.
- **Solution**: Use tanh or ReLU for hidden layers.
- **Value Range**: $(0, 1)$

---

c. Tanh (Hyperbolic Tangent):

- **Definition**:

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

- **Usage:** Hidden layers of neural networks.
- **Problems**:
  - Vanishing gradient problem for large inputs.
- **Solution**: Use ReLU or its variants.
- **Value Range**: $(-1, 1)$

---

d. ReLU (Rectified Linear Unit):

- **Definition**:

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

- **Usage**: Hidden layers of CNNs and deep networks.
- **Problems**:
  - **Dying ReLU**: Neurons output zero for negative inputs, leading to inactive neurons.
- **Solution**: Use Leaky ReLU or Parametric ReLU.
- **Value Range**: $[0, \infty)$

---

e. Leaky ReLU:

- **Definition**:

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

  (α is a small constant, e.g., 0.01.)

- **Usage**: Solves the "dying ReLU" problem in hidden layers.
- **Problems**:
  - Selection of α affects performance.
- **Solution**: Parametric ReLU (learnable α).

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

---

f. Softsign:

- **Definition**:

$$f(x) = \frac{x}{1 + |x|}$$

- **Usage**: Alternative to tanh.
- **Problems**:
  - Slower convergence compared to ReLU.
- **Solution**: Use ReLU for faster training.
- **Value Range**: $(-1, 1)$

---

g. Swish:

- **Definition**:

$$f(x) = x \cdot \sigma(x)$$

  ($\sigma(x)$ is the sigmoid function.)
- **Usage**: Advanced models like EfficientNet.
- **Problems**:
  - More computationally expensive than ReLU.
- **Solution**: Use only when better accuracy is required.
- **Value Range**: $(-\infty, \infty)$

---

3. Exponential Linear Units (ELUs)

ELUs provide smooth transitions and improve gradient flow for inputs less than zero.

Types of ELUs:

a. ELU (Exponential Linear Unit):

- **Definition**:

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

- **Usage**: Hidden layers to avoid vanishing gradients.
- **Problems**:
    - More computationally expensive than ReLU.
- **Solution**:
    - Use only when smoother gradients are necessary.
- **Value Range**:
    - $(-\alpha, \infty)$

---

b. Softmax:

- **Definition**:

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

- **Usage**: Output layer for multi-class classification.
- **Problems**:
    - May produce overconfident predictions.
- **Solution**:
    - Use techniques like temperature scaling for calibration.
- **Value Range**:
    - $(0, 1)$ (outputs sum to 1)

c. Softplus:

- **Definition**:

$$f(x) = \ln(1 + e^x)$$

- **Usage**: Smooth approximation of ReLU.
- **Problems**:
  - Computationally expensive.
- **Solution**:
  - Use simpler functions like ReLU when efficiency is critical.
- **Value Range**:
  - $(0, \infty)$

Comparison and Recommendations:

1. Use **ReLU** or **Leaky ReLU** in hidden layers for simplicity and efficiency.
2. Use **Sigmoid** or **Softmax** in the output layer for binary or multi-class classification.
3. Explore advanced functions like **Swish** for cutting-edge applications.