# Activation Functions in Neural Networks

This handout presents common activation functions with their formulas, conceptual motivation, pros/cons, computational aspects, and typical use cases.

Table 1: Common Activation Functions in Neural Networks

| **Activation** | **Formula / Concept** | **Motivation (Conceptual)** | **Advantages** | **Disadvantages** | **Computational** |
|---|---|---|---|---|---|
| Sigmoid | $f(x) = \frac{1}{1+e^{-x}}$ | Encodes "probabilistic firing" with smooth saturation; bounded output aids probability interpretation. | • Bounded in $(0, 1)$<br>• Probabilistic interpretation | • Vanishing gradients for large $\|x\|$<br>• Not zero-centered | • Uses `exp`; mod |
| Tanh | $f(x) = \tanh(x)$ | Center outputs at zero to speed optimization; antisymmetric. | • Zero-mean activations<br>• Smoother than sigmoid | • Still saturates $\Rightarrow$ vanishing gradients | • Similar cost to |
| ReLU | $f(x) = \max(0, x)$ | Mimics neuron firing only for positive signals; encourages sparsity. | • Simple and very fast<br>• Avoids saturation for $x > 0$ | • "Dying ReLU" for persistently negative inputs<br>• Unbounded output | • Branch/compa cheap) |
| Leaky ReLU | $f(x) = \max(\alpha x, x),\ \alpha \approx 10^{-2}$ | Preserve gradient flow for negative region to avoid dead units. | • Mitigates dying ReLU | • Small bias for $x < 0$ | • One multiply ReLU |
| PReLU | $f(x) = \max(\alpha x, x),\ \alpha$ learned | Learn slope for negatives from data to adapt nonlinearity. | • Adaptive capacity can improve accuracy | • Extra parameters; mild overfitting risk | • Slightly higher pute/memory |
| ELU | $f(x) = \begin{cases} x, & x > 0 \\ \alpha\big(e^{x} - 1\big), & x \le 0 \end{cases}$ | Push activations' mean toward zero; smooth negative branch. | • Faster convergence in some settings<br>• Avoids dead neurons | • Requires `exp`; sensitive to $\alpha$ | • Higher cost th |
| SELU | $\mathrm{SELU}(x) = \lambda \cdot \begin{cases} x, & x > 0 \\ \alpha\big(e^{x} - 1\big), & x \le 0 \end{cases}$ | Self-normalizing activations that keep layer activations near zero mean/unit variance. | • Stable activations without BatchNorm | • Requires LeCun normal init; architectural constraints | • Slightly costlie |
| Swish | $f(x) = x\,\sigma(x) = \frac{x}{1+e^{-x}}$ | Smooth gating of input magnitude; continuous ReLU alternative. | • Smooth gradients; strong empirical results | • Slower than ReLU; requires sigmoid | • Multiply + `ex` |
| Mish | $f(x) = x\,\tanh\big(\ln(1 + e^{x})\big)$ | Smooth, robust ReLU-like activation with improved gradient flow. | • Good stability and propagation | • Higher cost; less standardized | • Uses tanh and |
| GELU | $f(x) = x\,\Phi(x) \approx \frac{x}{2}\left(1 + \mathrm{erf}\left(\frac{x}{\sqrt{2}}\right)\right)$ | Probabilistic "gating by magnitude"; smooth ReLU-like behavior. | • Smooth; strong results in Transformers | • Slightly slower; derivative more complex | • Uses erf/norm |
| Softmax | $f_i(\mathbf{x}) = \frac{e^{x_i}}{\sum_j e^{x_j}}$ | Convert logits to a categorical probability distribution. | • Interpretable probabilities that sum to 1 | • Sensitive to large logits; numerical overflow | • Vector `exp` + |
| Linear | $f(x) = x$ | Preserve numeric information without distortion. | • Simple; suitable for regression | • No nonlinearity $\Rightarrow$ limited expressivity | • Minimal cost ( mapping) |

**Pedagogical Notes.** Nonlinearity is essential (without it, deep stacks collapse to an affine map). ReLU-family dominates CNN/MLP hidden layers; GELU/Swish are common in Transformers; tanh/sigmoid remain in RNN gates; SELU expects LeCun-normal initialization. Output activations depend on the task: Linear (regression), Sigmoid (binary), Softmax (multiclass). BatchNorm and initialization strongly interact with activation choice.