# Activation Functions and Learnable Parameters

In neural networks, **activation functions** introduce non-linearity after layers such as convolution or dense (fully connected) layers. Most of these functions do **not** have learnable parameters—they simply apply a fixed transformation to the input.

---

## Standard Activation Functions (No Learnable Parameters)

These functions operate element-wise and **do not involve training** or weights:

| Activation Function | Formula | Learnable Parameters | Notes |
|---|---|---|---|
| **ReLU** | $\max(0, x)$ | No | Most common activation |
| **Sigmoid** | $\frac{1}{1+e^{-x}}$ | No | Used in binary classification |
| **Tanh** | $\tanh(x)$ | No | Smooth zero-centered function |
| **Softmax** | $\frac{e^{x_i}}{\sum e^{x_j}}$ | No | Used in multi-class output layers |
| **GELU** | $x \cdot \Phi(x)$ (Gaussian approx) | No | Used in Transformers and BERT |
| **SELU** | Scaled ELU | No | Used in self-normalizing networks |
| **ELU** | $\begin{cases} x & x > 0 \\ \alpha(e^x - 1) & x \leq 0 \end{cases}$ | No | Smoother than ReLU below 0 |

These are **pure functions**—they don't adapt or change during training.

---

## Activation Functions with Learnable Parameters

Some specialized activations include **trainable elements**, allowing the network to adapt their behavior:

| Activation Function | Formula | Learnable Parameters | Description |
|---|---|---|---|
| **PReLU** | $\begin{cases} x & x \geq 0 \\ ax & x < 0 \end{cases}$ | Yes (slope $a$) | Slope for negative values is learned |
| **Learnable Swish** | $x \cdot \text{sigmoid}(\beta x)$ | Yes (slope $\beta$) | Variant of Swish with trainable slope |
| **AReLU / APL** | Adaptive piecewise-linear units | Yes | Allows complex learned activation shapes |
| **Soft Adapt. Act.** | Adaptive parametric functions (e.g. spline-based) | Yes | Custom trainable activation functions |

> These are **less commonly used**, but can improve model performance in certain scenarios.

---

## Summary

- **Most activation functions have no learnable parameters** — they simply apply a fixed transformation to the input.
- A few specialized or experimental activations (like **PReLU**, **Learnable Swish**, and **APL**) **do** include parameters that are learned during training.
- **You don’t count ReLU, tanh, sigmoid, etc., when summing model parameters**.

---

## When to Use Learnable Activations?

- Use **ReLU** or **GELU** in most architectures — they're fast, reliable, and non-parametric.
- Use **PReLU** if:

    - You suspect that the slope of the negative part needs to adapt.
    - You're trying to avoid “dead” neurons (especially in deep CNNs).
- Use **adaptive activations** in research or experimentation, not in standard production models unless well-tested.

---