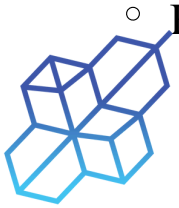
Activation Functions

Activation Function	Use Case	Pros	Cons
Sigmoid	Binary classification, output layers	Outputs between 0 and 1; good for probabilities	Vanishing gradient problem
Tanh	Hidden layers, regression tasks	Outputs between -1 and 1; zero-centered	Vanishing gradient problem
ReLU (Rectified Linear Unit)	Hidden layers, convolutional neural networks (CNNs)	Computationally efficient; reduces vanishing gradient	Can cause dying ReLUs (neurons stop learning)
Leaky ReLU	Hidden layers, avoiding dying ReLUs	Prevents dying ReLUs by allowing small gradients	Introduces a small slope for negative inputs
ELU (Exponential Linear Unit)	Hidden layers, deep neural networks	Smooths the transition, faster and reliable convergence	More computationally expensive
SELU (Scaled Exponential Linear Unit)	Self-normalizing networks, hidden layers	Self-normalizes outputs; robust training	Requires specific initialization and dropout scaling
Softmax	Output layers for multiclass classification	Converts logits to probabilities; good for classification	Can be computationally intensive for many classes

Softplus	Hidden layers, smooth activation	Smooth approximation of ReLU; always positive	Computationally more complex than ReLU
Softsign	Hidden layers, alternative to Tanh	Bounded outputs; reduces vanishing gradient problem	Slower convergence compared to ReLU and Tanh

Descriptions

1. **Sigmoid**: Maps the input to a value between 0 and 1. Often used in the output layer of binary classification problems.
   - PyTorch Syntax: torch.nn.Sigmoid()
2. **Tanh**: Maps the input to a value between -1 and 1. Often used in hidden layers and can be seen as a scaled version of the sigmoid function.
   - PyTorch Syntax: torch.nn.Tanh()
3. **ReLU (Rectified Linear Unit)**: Applies the rectifier function, which outputs the input directly if it is positive; otherwise, it outputs zero. Commonly used in hidden layers of neural networks.
   - PyTorch Syntax: torch.nn.ReLU()
4. **Leaky ReLU**: A variation of ReLU that allows a small, non-zero gradient when the input is negative, which helps to avoid dying ReLUs.
   - PyTorch Syntax: torch.nn.LeakyReLU()
5. **ELU (Exponential Linear Unit)**: An activation function that tends to converge faster and more reliably than ReLU by smoothing the transition from negative to positive inputs.
   - PyTorch Syntax: torch.nn.ELU()
6. **SELU (Scaled Exponential Linear Unit)**: A self-normalizing activation function that automatically scales the output to have a mean of 0 and standard deviation of 1.
   - PyTorch Syntax: torch.nn.SELU()



7. **Softmax**: Converts the input values to probabilities, summing to 1, often used in the output layer of multiclass classification problems.
    - PyTorch Syntax: torch.nn.Softmax()
8. **Softplus**: Smooth approximation of ReLU, used to ensure the output is always positive.
    - PyTorch Syntax: torch.nn.Softplus()
9. **Softsign**: Maps the input to a value between -1 and 1, similar to Tanh but with a different scaling.
    - PyTorch Syntax: torch.nn.Softsign()