======================================================================

**Activation functions**

======================================================================

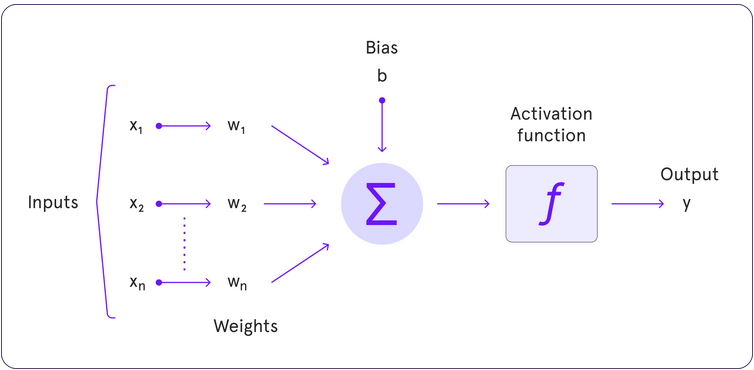
An activation function is a mathematical function applied to each neuron in a neural network. It determines whether a neuron should be activated based on its input. This introduces non-linearity, allowing neural networks to learn complex patterns in data.

Without this non-linearity feature, a neural network would behave like a linear regression model, no matter how many layers it has.

Non-linearity means that the relationship between input and output is not a straight line. In simple terms, the output does not change proportionally with the input.

**Why Do We Need Activation Functions?**

* I**ntroduces non-linearity** → Without activation functions, a deep neural network is just a linear function (like a simple regression).
* **Enables deep networks to learn complex patterns** → Helps the network capture relationships in the data.
* **Prevents vanishing/exploding gradients** → Some activation functions help in stabilizing training.

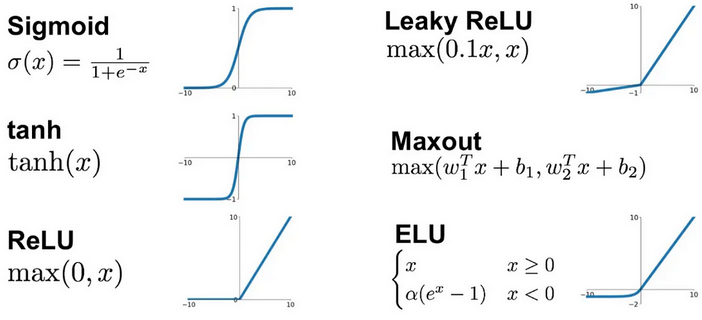


**Why is Non-Linearity Important in Neural Networks?**

Neural networks consist of neurons that operate using **weights**, **biases**, and **activation functions**.

In the learning process, these weights and biases are updated based on the error produced at the output—a process known as **backpropagation**. Activation functions enable backpropagation by providing gradients that are essential for updating the weights and biases.

Without non-linearity, even deep networks would be limited to solving only simple, linearly separable problems. Activation functions empower neural networks to model highly complex data distributions and solve advanced deep learning tasks. Adding non-linear activation functions introduce flexibility and enable the network to learn more complex and abstract patterns from data.



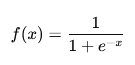
An ideal Activation function should be:

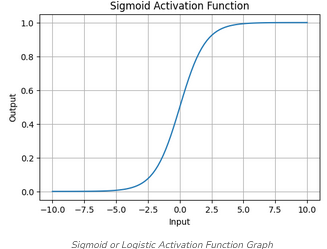
1. Non-Linear: To identify the non-linear patterns.
2. Differentiable (not mandatory): Should be able to calculate the derivatives, so that we can apply gradient descent and back propagation.
3. Computationally in-expensive, so that it would be fast.
4. Zero centred – (normalized, mean = 0) – Converges fast and performance would be better.
5. Non-Saturating – Should not squeeze the inputs into a range. Ex: Sigmoid always squeezes the input into (0, 1)

======================================================================

**Sigmoid activation function**

======================================================================



 Output between –[0,1]

Advantages:

1. Good for Binary classification (e.g., logistic regression), gives the probability as an output.
2. Non-linear – which is a good option for activation functions.
3. Differentiable – works well for back propagation.

Dis-advantages:

1. Saturating function – Squeezes the input values into a range. Which causes vanishing gradient problem. (i.e. in back propagation – no update would take place and training would be stopped because of small gradients). **This is the main reason why we don’t use Sigmoid function in the hidden layers as activation function.**
2. Non zero centred. - The mean of the (output of activation function) is not normalized (non-zero) – this causes slowness in the convergence.
3. Computationally expensive as it involves exponential.

So mostly sigmoid would be used only in the output layer while working in the binary classification problems.

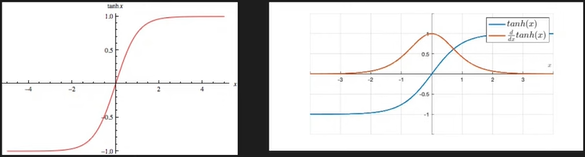
======================================================================

**Tanh (Hyperbolic Tangent)**

======================================================================



Output: [-1, 1]



Advantages:

1. Non-linear – Can capture non linear patterns in the data.
2. Differentiable – Derivatives can be applied.
3. Zero centred - +ve and -ve both – Training would be speeder than Sigmoid.

Disadvantages:

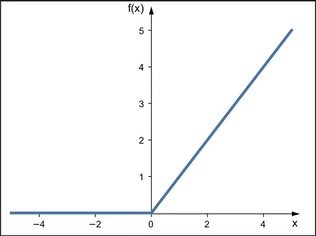
1. Saturating function – Squeezes the input- causes vanishing gradient.
2. Computationally expensive (as it involves exponents calculation)

======================================================================

**Relu (Rectified Linear Unit) Activation Function**

======================================================================





Advantages:

1. Non-linear
2. Not saturated in the positive region. (not squeezing) - Solves vanishing gradient issue.
3. Computationally in-expensive.
4. Convergence is faster than sigmoid and Tanh.

Disadvantages:

1. Not completely differentiable.
2. Not Zero Centred.

Best activation function and mostly used among others (though it has some drawbacks)

======================================================================

**Dying Relu:**

Dying ReLU is a problem in deep learning where some neurons in a network always output zero and stop learning. This happens because the ReLU (Rectified Linear Unit) activation function outputs 0 for all negative inputs.

**Why Does It Happen?**

The **ReLU function** is defined as:



* If x > 0, ReLU outputs x (positive value).
* If x ≤ 0, ReLU outputs 0 (neuron is inactive).

During training, if a neuron's weights are updated in such a way that it always produces negative inputs, it will always output 0 and never activate again. These neurons are considered "dead" and stop contributing to learning.

**Effects of Dying ReLU**

* **Reduces network capacity** → Fewer active neurons mean lower learning ability.
* **Slows down training** → A large portion of neurons become inactive.
* **Happens more with large negative weight updates** → Can occur in deep networks.

**How to Fix Dying ReLU?**

**Use other ReLU variants**

* such as Leaky ReLU, Parametric ReLU (PReLU), ELU (Exponential Linear Unit)

**Use Small Learning Rates**

* Large weight updates can push neurons into the dead zone.
* A smaller learning rate prevents drastic weight changes.

**Use Batch Normalization**

* Helps maintain a balanced distribution of activations.
* Prevents neurons from getting stuck with negative values

-------------------------------------------------------------------------------------------------------------

**Variants of ReLU function:**

**Linear:**

* Leaky ReLU
* Parametric ReLU

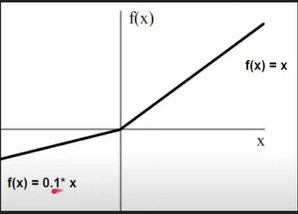
**No-linear:**

* ELU
* SELU

--------------------------------------------------------------------------------------------------------------

**Leaky ReLU (Improved ReLU)**

 ex: Max (0.01x, x)

Alpha is usually 0.01

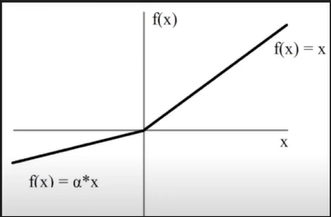
Advantages:

* 1. Non saturated function.
  2. Easily computed
  3. Solves dying ReLU problem (small slope for negative values)
  4. Close to Zero Centred.

--------------------------------------------------------------------------------------------------------------

**Parametric ReLU (PReLU)**

Similar to Leaky ReLU, but α (alpha) is learned during training.

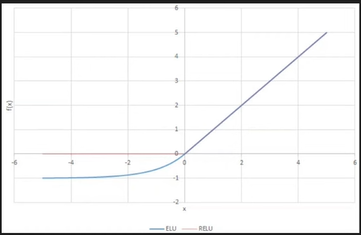
****

Depending on the training alpha value would be adjusted.

**--------------------------------------------------------------------------------------------------------------ELU: (Exponential Linear Unit)**

ELU avoids zero gradients by using an exponential function for negative values.

****

****

Performance is better than ReLU.

**Advantages:**

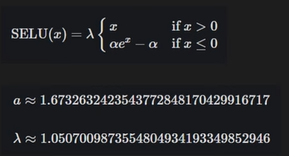
1. Zero centred – Convergence faster.
2. Better generalization than ReLU.
3. Always continuous and always differentiable.

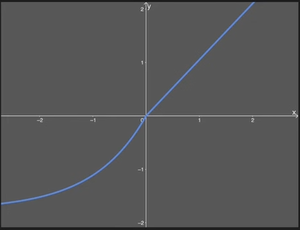
**Disadvantages:**

1. Computationally expensive.

**--------------------------------------------------------------------------------------------------------------**

**SELU (Scaled Exponential Linear Unit)**





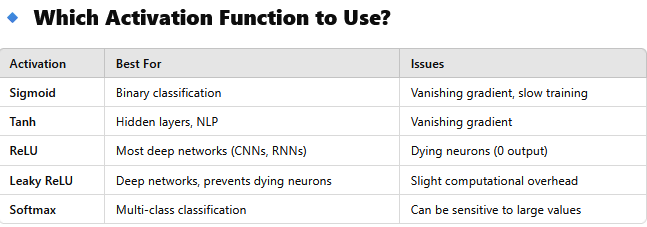
--------------------------------------------------------------------------------------------------------------

**Softmax (for Multi-Class Classification)**

****

🔹 Outputs probabilities for multi-class classification  
🔹 Ensures sum of outputs = 1

--------------------------------------------------------------------------------------------------------------



**Summary**

* Activation functions introduce non-linearity.
* ReLU and its variants (Leaky ReLU) are most commonly used.
* Sigmoid and Tanh suffer from the vanishing gradient problem.
* Softmax is ideal for multi-class classification.

===========================================================