# Types of Activation Function

## Binary Step Function

This is a very intuitive activation function which only enables or disables artificial neuron.

It can only produce two values 0 or 1.

If it is 0 it means neuron has not fired, if it is 1 it means neuron has fired. It produces output 1 only when the sum of dot products of inputs & weights is more than a threshold value, else it will be 0.

**Limitations:**

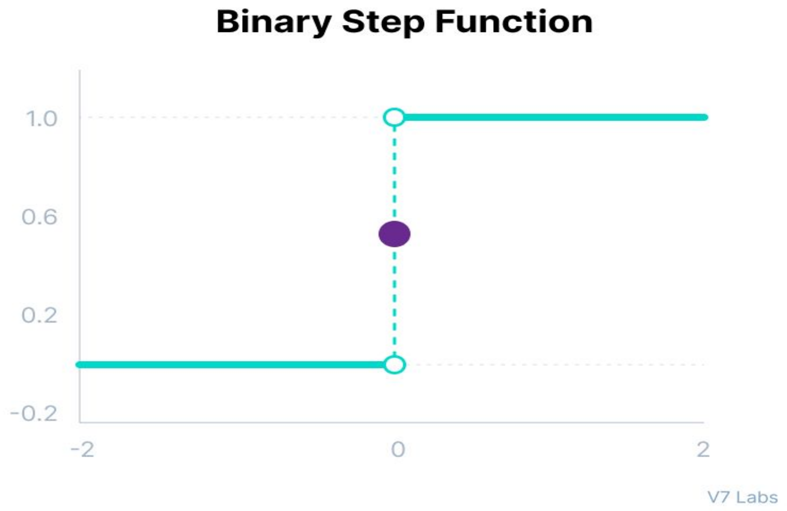
- It cannot provide multi-value outputs—for example, it cannot be used for multi-class classification problems.
- The gradient of the step function is zero, which causes a hindrance in the backpropagation process.

## Binary Step Function

Mathematically it can be represented as:

$$\textit{Binary step}$$

$$f(x) = \begin{cases} 0 & for\ x < 0 \\ 1 & for\ x \geqslant 0 \end{cases}$$



## Linear Activation Function

The linear activation function, also known as "no activation," or "identity function", is where the activation is proportional to the input.

**Two major problems :**

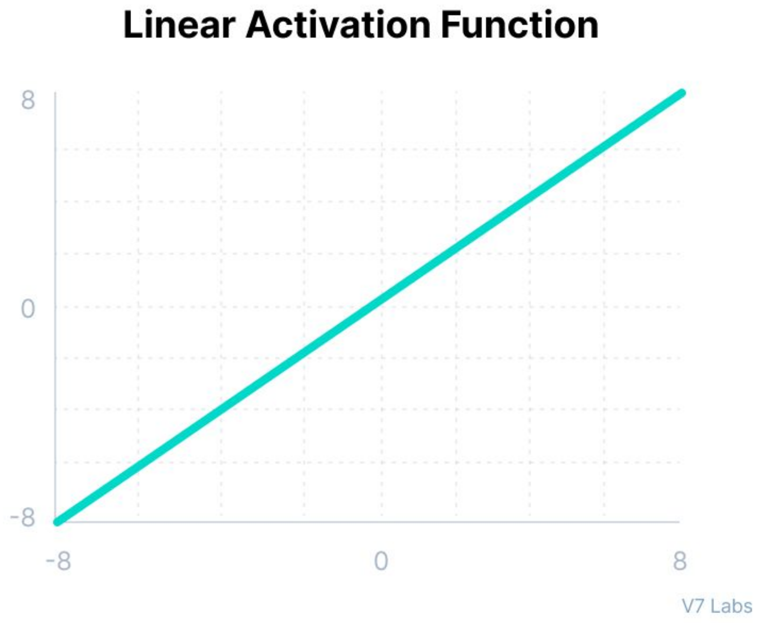
- It’s not possible to use backpropagation as the derivative of the function is a constant and has no relation to the input x.
- All layers of the neural network will collapse into one if a linear activation function is used. No matter the number of layers in the neural network, the last layer will still be a linear function of the first layer. So, essentially, a linear activation function turns the neural network into just one layer.

## Linear Activation Function

Mathematically it can be represented as:

$$Linear$$

$$f(x) = x$$



## Non-Linear Activation Function

Non-linear activation functions solve the limitations of linear activation functions:

- They allow backpropagation because now the derivative function would be related to the input, and it’s possible to go back and understand which weights in the input neurons can provide a better prediction.
- They allow the stacking of multiple layers of neurons as the output would now be a non-linear combination of input passed through multiple layers. Any output can be represented as a functional computation in a neural network.

Different types of non-linear activation functions are discussed in next slides.

## 1. Sigmoid/Logistic Activation Function

- The sigmoid function is S-Shaped and is smooth .
- The range of output is between 0 and 1 which is usually interpreted as probability if used in the output layer.
- This function is both non-linear and is differentiable.
- Suffers from Vanishing Gradient Problem.

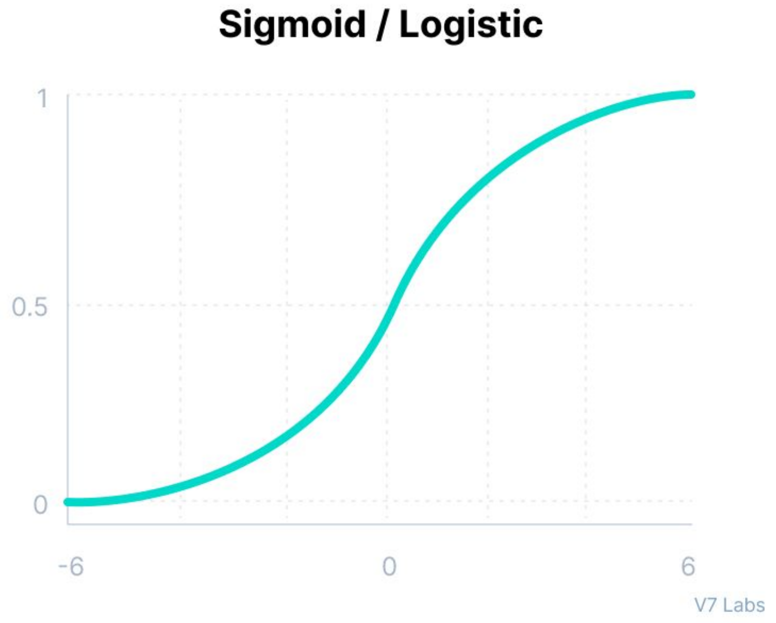
The output of the logistic function is not symmetric around zero. So the output of all the neurons will be of the same sign. This makes the training of the neural network more difficult and unstable.

## 1. Sigmoid/Logistic Activation Function

Mathematically it can be represented as:

*Sigmoid / Logistic*

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



## 2. Tanh Function

Tanh function is very similar to the sigmoid/logistic activation function, and even has the same S-shape with the difference in output range of -1 to 1.

In Tanh, the larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to -1.0.

The output of the tanh activation function is Zero centered; hence we can easily map the output values as strongly negative, neutral, or strongly positive.

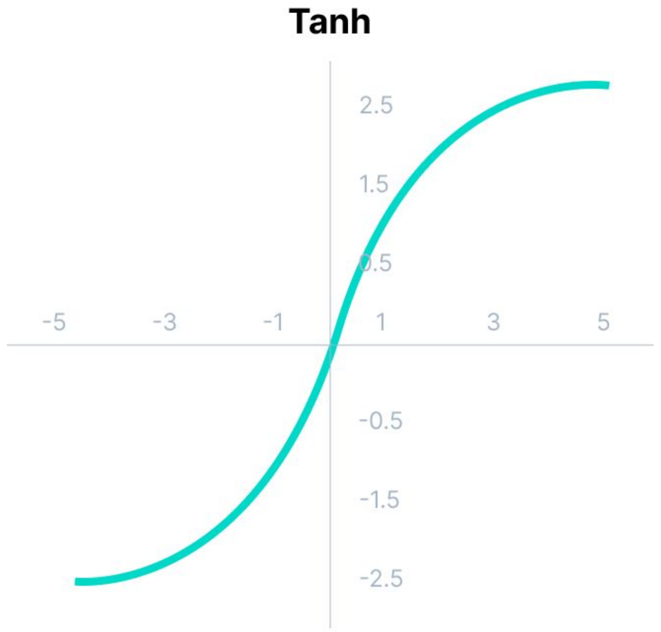
**Although both sigmoid and tanh face vanishing gradient issue, tanh is zero centered, and the gradients are not restricted to move in a certain direction. Therefore, in practice, tanh nonlinearity is always preferred to sigmoid nonlinearity.**

## 2. Tanh Function

Mathematically it can be represented as:

$$Tanh$$

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



## 3. ReLU Function

- ReLU stands for Rectified Linear Unit.
- Although it gives an impression of a linear function, ReLU has a derivative function and allows for backpropagation while simultaneously making it computationally efficient.
- The main catch here is that the ReLU function does not activate all the neurons at the same time.
- The neurons will only be deactivated if the output of the linear transformation is less than 0.
- f(x) is zero when x is less than zero and f(x) is equal to x when x is above or equal to zero.

## 3. ReLU Function

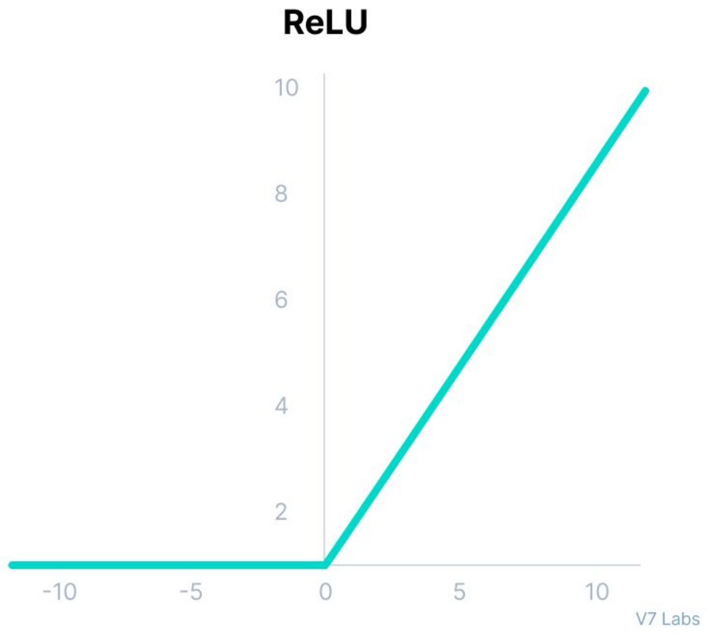
But the issue is that all the negative values become zero immediately which decreases the ability of the model to fit or train from the data properly. That means any negative input given to the ReLU activation function turns the value into zero immediately in the graph, which in turns affects the resulting graph by not mapping the negative values appropriately. This is called **Dying ReLU Problem.**

## 3. ReLU Function

Mathematically it can be represented as:

$$ReLU$$

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



## 4. Leaky ReLU Function

Leaky ReLU is an improved version of ReLU function to solve the Dying ReLU problem as it has a small positive slope in the negative area.

The advantages of Leaky ReLU are same as that of ReLU, in addition to the fact that it does enable backpropagation, even for negative input values.
By making this minor modification for negative input values, the gradient of the left side of the graph comes out to be a non-zero value. Therefore, we would no longer encounter dead neurons in that region.

**Limitations :**

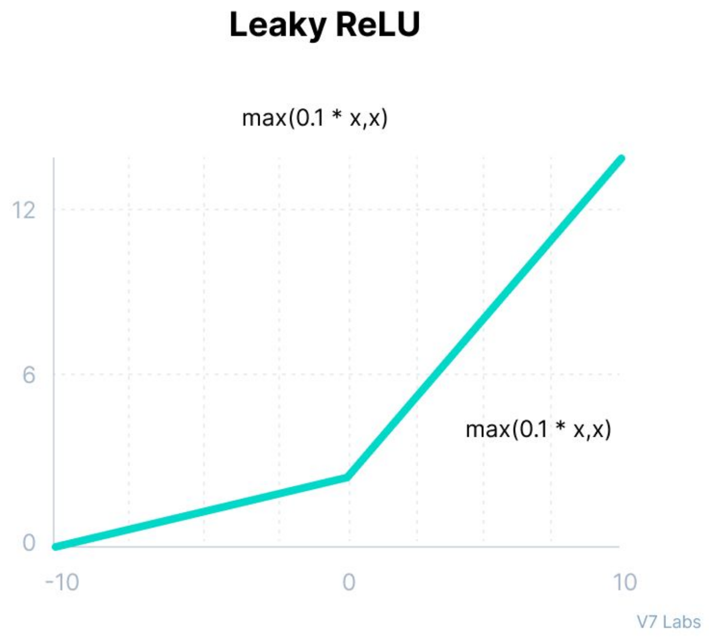
- The predictions may not be consistent for negative input values.
- The gradient for negative values is a small value that makes the learning of model parameters time-consuming.

## 4. Leaky ReLU Function

Mathematically it can be represented as:

*Leaky ReLU*

$$f(x) = max\ (0.1x, x)$$



## 5. Softmax Function

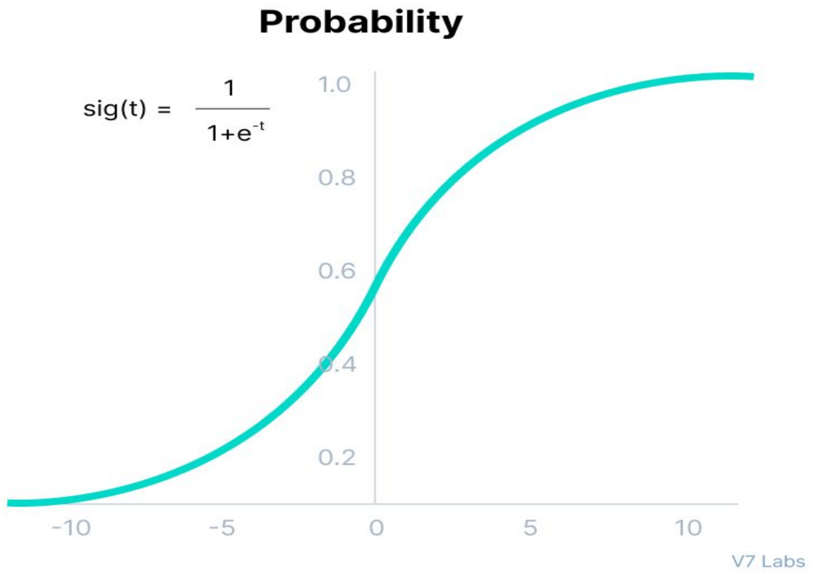
- Softmax function is described as a combination of multiple sigmoids.
- It calculates the relative probabilities. Similar to the sigmoid/logistic activation function, the SoftMax function returns the probability of each class.
- Used for multi class classification.
- Softmax function converts real values into probabilities.

## 5. Softmax Function

**Softmax**

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\Sigma_j \exp(z_j)}$$

V7 La



## Vanishing Gradient

- Like the sigmoid function, certain activation functions squish an ample input space into a small output space between 0 and 1.
- Therefore, a large change in the input of the sigmoid function will cause a small change in the output. Hence, the derivative becomes small. For shallow networks with only a few layers that use these activations, this isn’t a big problem.
- However, when more layers are used, it can cause the gradient to be too small for training to work effectively.

## Exploding Gradient

- Exploding gradients are problems where significant error gradients accumulate and result in very large updates to neural network model weights during training.
- An unstable network can result when there are exploding gradients, and the learning cannot be completed.

## How to choose right activation function?

- Sigmoid functions and their combinations generally work better in the case of classifiers.
- Sigmoid and Tanh functions should be avoided in hidden layers as it might lead to a vanishing gradient problem.
- Sigmoid and softmax functions are now generally used in the output layer in case of binary and multi-class classification respectively.
- ReLU is now popularly used in the hidden layer as it is computationally faster and does not suffer from vanishing gradient problems.
- ReLU can suffer from dying neurons problem, to avoid this Leaky ReLU is used sometimes.

## How to choose right activation function?

Activation function for output layer based on the type of prediction problem:

- **Regression** - Linear Activation Function
- **Binary Classification**—Sigmoid/Logistic Activation Function
- **Multiclass Classification**—Softmax
- **Multilabel Classification**—Sigmoid

## How to choose right activation function?

The activation function used in hidden layers is typically chosen based on the type of neural network architecture:

- **Convolutional Neural Network (CNN):** ReLU activation function.
- **Recurrent Neural Network:** Tanh and/or Sigmoid activation function.

You can google to know more about other types of activation functions and their use cases.