ACTIVATION FUNCTIONS

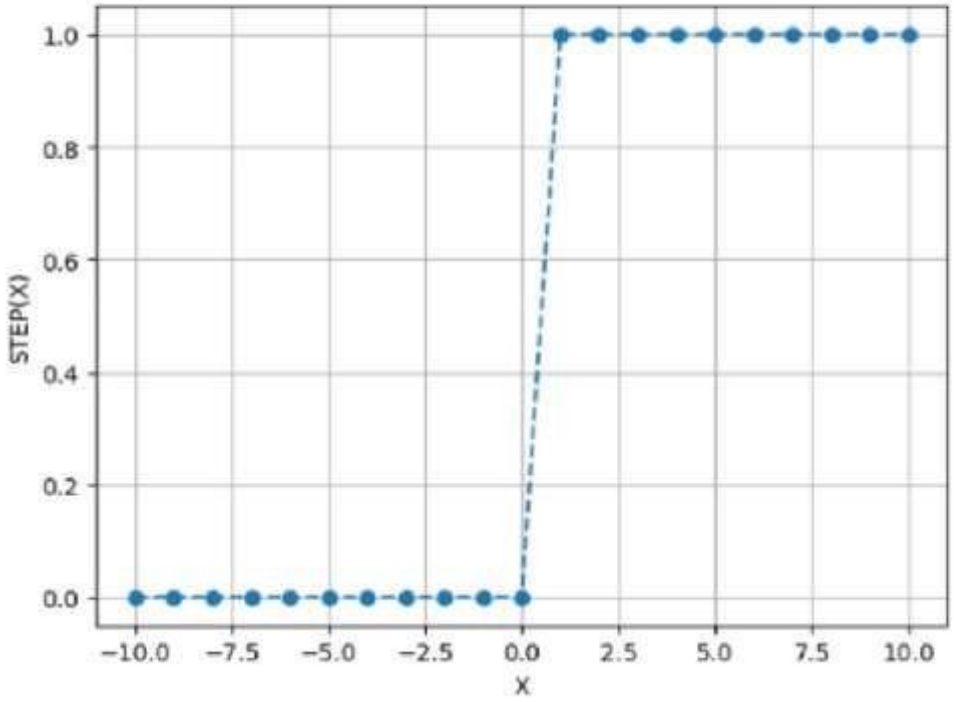
An Activation Function **decides whether a neuron should be activated or not**. This means that it will decide whether the neuron's input to the network is important or not in the process of prediction using simpler mathematical operations.

STEP FUNCTION

The step function, also known as the Heaviside step function, is a simple activation function commonly used in neural networks. It is defined mathematically as follows:

$$H(x) = \begin{cases} 0, & if\ x < 0 \\ \&1, & if\ x \geq 0 \end{cases}$$

```
y = list(map(lambda n: 1 if n>0.5 else 0, x))
plot_graph(y,"STEP(X)")
```



ADVANTAGES:

- Simple, easy to implement and interpret.
- It is a binary function.
- It is easy to interpret and understand, which makes it useful for educational purposes.

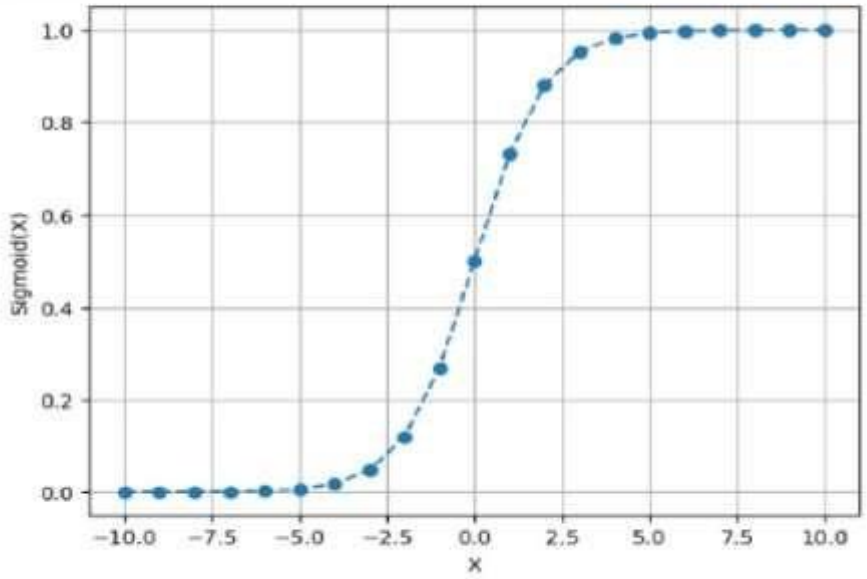
DISADVANTAGES:

- Output is not continuous and can't be differentiated. Limit to binary classification problem.
- The step function is not differentiable at x=0, which can cause problems when training neural networks using gradient descent algorithms.
- The step function can suffer from the problem of vanishing gradients, where the gradients of the function become too small to be useful for training the network.

SIGMOID FUNCTION

The sigmoid function is a mathematical function that maps any input to a value between 0 and 1. The most commonly used sigmoid function is the logistic function, which is defined as: f(x) = 1 / (1 + e^(-x)) where x is the input to the function.

```
[5]: y = 1 / (1 + np.exp(-x))
     plot_graph(y, "Sigmoid(X)")
```



ADVANTAGES:

- Output is bounded between 0 & 1 which makes it suitable for binary classification problem.
- It is differentiable and use in backpropagation.
- They are widely used in neural networks to map the output of a neuron to a probability distribution.

DISADVANTAGES:

- Can suffer from vanishing gradient and prone to saturation for extreme values.
- Not zero-centered.
- The gradient of the sigmoid function becomes very small for large values of x, which can lead to slow convergence in optimization algorithms.

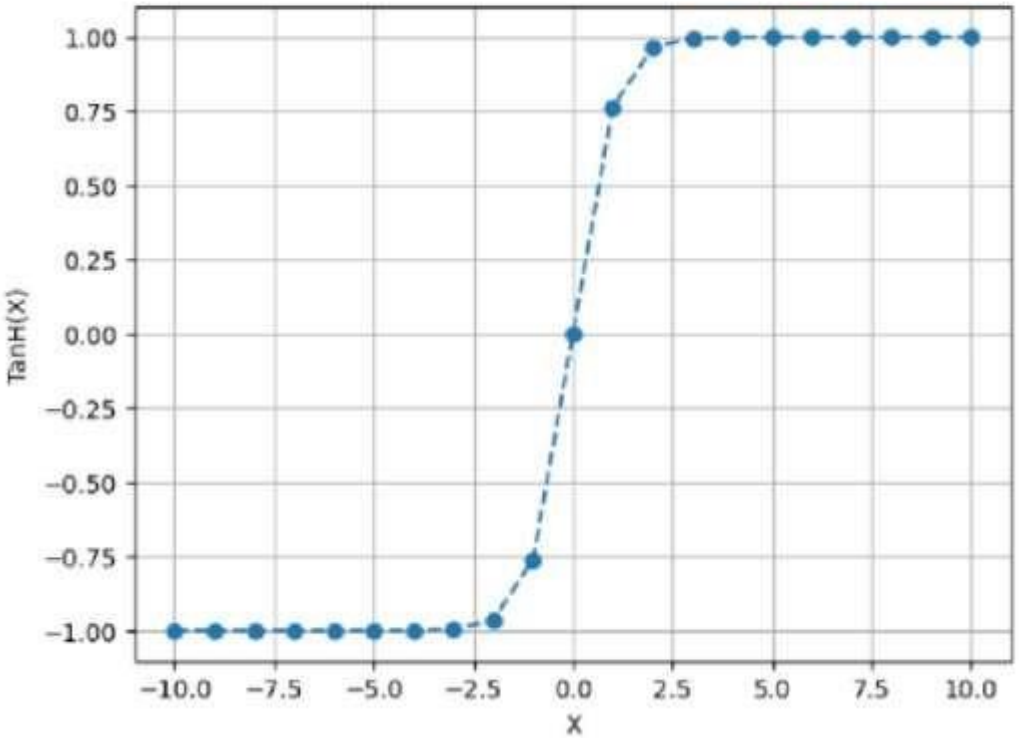
TANH FUNCTION

A mathematical function called the hyperbolic tangent function (tanh) converts input values into output values between -1 and 1. The equation is as follows:

tanh(x) = (e^x - e^(-x)) / (e^x + e^(-x))

where e is the mathematical constant approximately equal to 2.71828, and x is the input value.

```
y = (np.exp(2*x) - 1) / (np.exp(2*x) + 1)
plot_graph(y, "TanH(X)")
```



ADVANTAGES:

- Same to sigmoid but zero-centered.
- Outputs are bound between -1 & 1.
- I t is a zero-centered function, which means that its outputs are centered around zero. This can help in preventing vanishing gradients during the training of deep neural networks.

DISADVANTAGES:

- Like sigmoid, it can suffer from vanishing gradients.
- Tanh is not monotonic, which means that its derivative is not always positive or negative.
- The output of tanh is not sparse, which means that it can be less efficient than other activation functions in terms of memory and computation requirements.

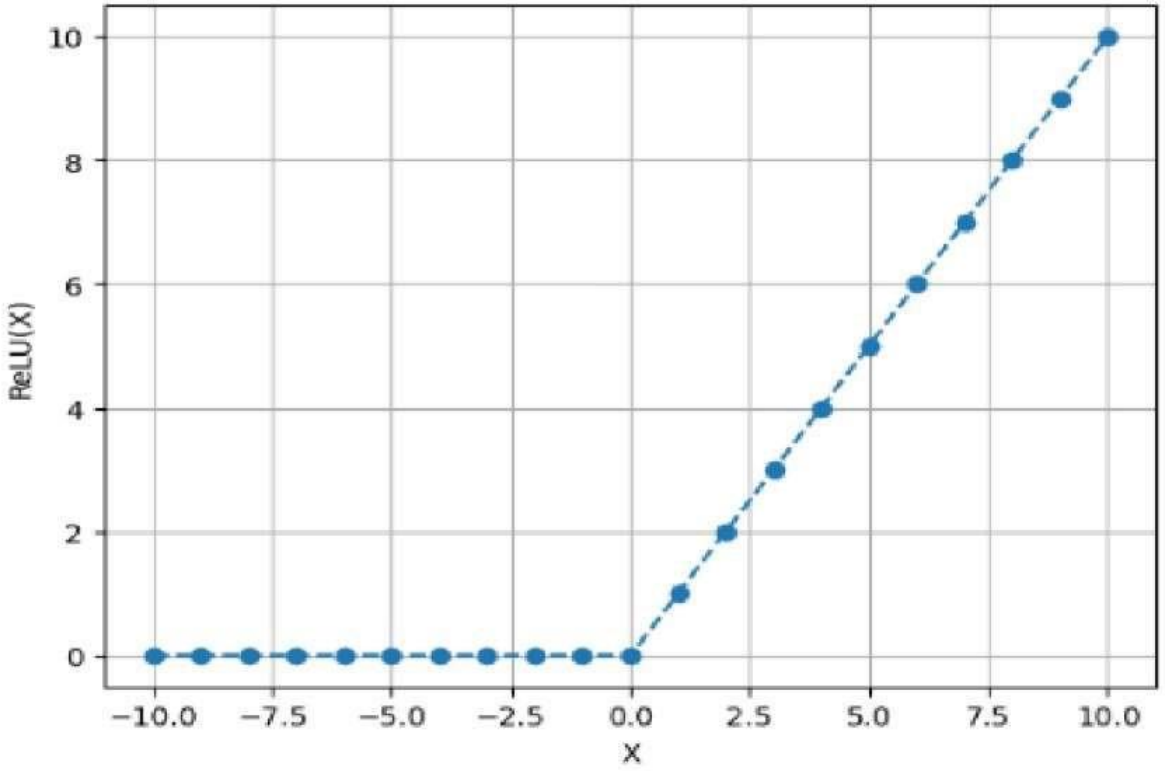
RELU FUNCTION

The ReLU (Rectified Linear Unit) function is a commonly used activation function in neural networks. It is defined as:

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

where x is the input to the function.

```
y = list(map(lambda a: a if a>=0 else 0, x))
plot_graph(y,"ReLU(X)")
```



ADVANTAGES:

- Simple, computationally efficient and has been shown to work well in practice.
- It avoids the vanishing gradient problem.
- It has been shown to work well in many types of neural networks, including deep neural networks.
- The ReLU function has a sparse output, which can help prevent overfitting.

DISADVANTAGES:

- Not zero-centered and can suffer from the “dying ReLU” problem where the gradient becomes zero for negative inputs and stop training.
- The ReLU function is not symmetric, which can make it difficult to use in certain types of neural networks.

ELU FUNCTION

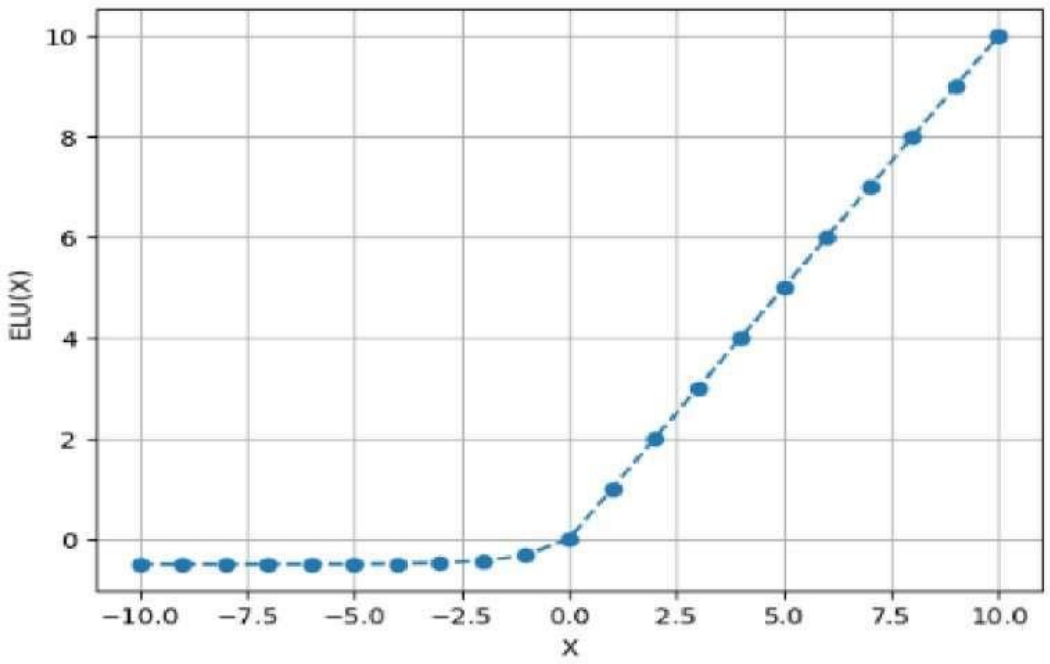
The ELU (Exponential Linear Unit) step function is an activation function used in artificial neural networks. It is a variant of the ReLU (Rectified Linear Unit) function and is defined as:

ELU(x) = { x if x >= 0, alpha * (exp(x) - 1) if x < 0 }

where alpha is a hyperparameter that determines the negative saturation value.

```
from math import exp
alpha = 0.5

elu = lambda x, alpha: x if x > 0 else alpha * (np.exp(x) - 1)
y = [elu(x_i, alpha) for x_i in x]
plot_graph(y,"ELU(X)")
```



ADVANTAGES:

- Avoids the "dying ReLU" issue and is able to generate negative output.
- The ELU function has a continuous derivative, which helps avoid the vanishing gradient problem that can occur with other activation functions like the sigmoid function.
- The ELU function has been shown to help networks converge faster than other activation functions.

DISADVANTAGES:

- Somewhat more expensive to compute than ReLU.
- The ELU function can be unstable for large negative inputs, which can lead to numerical instability in the network.
- The ELU function is not as well-known as other activation functions like the sigmoid and ReLU functions.

SELU FUNCTION

The SELu (Scaled Exponential Linear Unit) activation function is a type of activation function used in neural networks. Unlike other activation functions, such as ReLU, which have a hard cutoff at zero, SELu has a smooth, continuous curve.

The formula for the SELu activation function is:

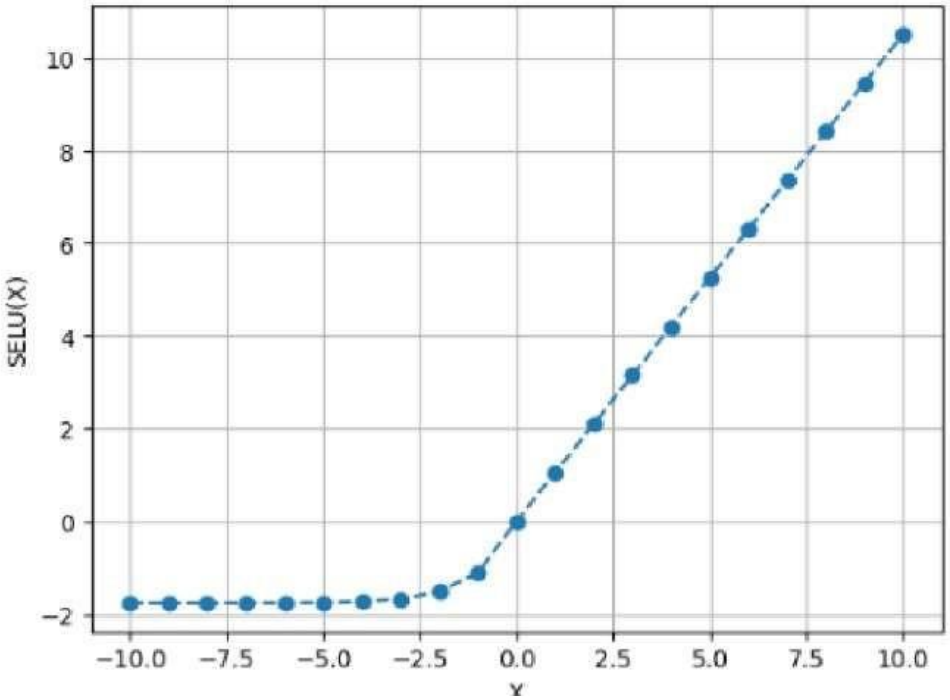
$$f(x) = 1.0507 * (e^x - 1), x < 0$$

$$f(x) = x, x >= 0$$

```
alpha = 1.67326
scale = 1.0507

selu = lambda x, alpha, scale: scale * (x if x > 0 else alpha * (np.exp(x) - 1))
y = [selu(x_i, alpha, scale) for x_i in x]

plot_graph(y,"SELU(X)")
```



ADVANTAGES:

- Intended to achieve self-normalization in deep neural networks, resulting in more rapid convergence and enhanced performance.
- Having a zero-centered and smooth curve.
- SELu has been shown to perform better than other activation functions, such as ReLU and tanh, on a variety of deep learning tasks, including image classification and speech recognition.

DISADVANTAGES:

- To achieve self-normalization, weights and biases must be properly initialized.
- The SELu activation function is computationally more expensive than other activation functions, such as ReLU, because it involves exponentiation.
- SELu may not work well on all types of data. While it has been shown to work well on many deep learning tasks, it may not be the best choice for all types of data, and some experimentation may be necessary to determine the best activation function for a particular task.

CONCLUSION

Hence, there is no universally superior activation function for all types of problems. Best practice is to experiment with different activation functions and choose the one that works best for your particular problem and architecture. But, ReLU is a good default choice for many problems, and ELU and SELU are good alternatives when negative inputs are present or when deeper architecture is used. The ReLU is the most used activation function in the world right now.Since, it is used in almost all the convolutional neural networks or deep learning.