## *Step Function*

The step function accepts an input and, depending on whether the input is larger than or equal to a threshold value, returns a binary output of 0 or 1. The step function's graph is a straight line that changes from 0 to 1 at the threshold value. The output is 0 if the threshold value is less than it, and 1 if the threshold value is greater. The step function is discontinuous, which makes it difficult to work with in some situations because it jumps sharply at the threshold value.

## *Sigmoid Function*

The sigmoid activation function is helpful for tasks where the output of a model needs to be understood as a probability. It maps any input to a value between 0 and 1. As can be seen from the graph, the sigmoid function is a continuous, smooth curve that rises steadily over time from 0 to 1. Its gradual slope near the middle makes it an excellent option for representing probability. Deep neural networks that use this function may find it difficult to train due to the "vanishing gradient" problem, which occurs as the gradient of the function approaches 0 as the input goes away from the inflection point.

## *Tanh Function*

Tanh activation function converts any input to a number between -1 and 1. This property makes it helpful for situations where the output of a model must be symmetrical about 0. The tanh function graph is a continuous curve with a rounded ‘S’ shape. It approaches -1 as the input near negative infinity, approaches 1 as the input approaches positive infinity, and crosses the 0 line at its inflection point. Because of the function's gradual slope around its center, symmetrical data can be represented using it effectively. Yet it may also suffer from the "vanishing gradient" issue, when the slope of the function approaches 0 as the input goes away from the inflection point, just like the sigmoid function.

## *ReLU Function*

The Rectified Linear Unit (ReLU) activation function transfers any input less than 0 to 0 and any input equal to or higher than 0 to the input value itself. The ReLU function has a straight line with an origin on its graph (0, 0). A slope of 1 for all positive input values and 0 for all negative input values characterize this piecewise linear function. Because of this, deep learning models often use it as an activation function that is computationally efficient to compute. However, it can also suffer from the "dying ReLU" problem, where the output is always 0.

## *Leaky ReLU Function*

The traditional ReLU function has been modified with Leaky ReLU function in order to address the issue of "dying ReLU." Though extremely similar to the ReLU function, the Leaky ReLU

function has a modest, non-zero slope for negative input values. As a result, the model can continue to improve even when its output is negative.

## *ELU Function*

Standard ReLU function has some drawbacks, which the Exponential Linear Unit (ELU) activation function attempts to overcome. The ELU function's graph is a piecewise continuous function with an exponential curve for positive input values and a value of 0 for all other input values. The "dying ReLU" issue can be avoided by the ELU function being able to produce negative output values thanks to the exponential curve, which gives neurons with a slope of zero some gradient.

## *SELU Function*

The graph of the Scaled Exponential Linear Unit (SELU) activation function is an exponential curve for positive input values and a piecewise continuous function for all other input values. The scaling parameters alpha and lambda for the SELU function are both integers. When the inputs are negative, lambda regulates the scaling of the output, and when the inputs are positive, alpha controls the slope of the exponential curve.