Activation Function

An activation function is a function used in artificial neural networks which outputs a small value for small inputs, and a larger value if its inputs exceed a threshold. If the inputs are large enough, the activation function "fires", otherwise it does nothing. The role of the Activation Function is to derive output from a set of input values fed to a node or a layer. In other words, an activation function is like a gate that checks that an incoming value is greater than a critical number.

Activation functions are useful because they add non-linearities into neural networks, allowing the neural networks to learn powerful operations. If the activation functions were to be removed from a feedforward neural network, the entire network could be re-factored to a simple linear operation or matrix transformation on its input, and it would no longer be capable of performing complex tasks such as image recognition. Activation functions in computer science are inspired by the action potential in neuroscience. If the electrical potential between a neuron's interior and exterior exceeds a value called the action potential, the neuron undergoes a chain reaction which allows it to 'fire' and transmit a signal to neighboring neurons. The resultant sequence of activations, called a 'spike train', enables sensory neurons to transmit feeling from the fingers to the brain, and allows motor neurons to transmit instructions from the brain to the limbs.

Binary Step Function : Binary step function is a threshold-based activation function which means after a certain threshold neuron is activated and below the said threshold neuron is deactivated. It can be easily used at the output layer of a neural network for binary classification problems as the value of sigmoid function lies between 0 and 1 and output can be predicted by keeping some threshold value between 0 and 1.

The step function has the following mathematical definition:

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

$f(x) = 1$, if $x >= 0$.

It has a Smooth gradient, preventing jumps in output values. It is computationally effective as it involves simpler mathematical operations than sigmoid and tanh. However, The gradient of the step function is zero, which causes a hindrance in the backpropagation process. It cannot provide multi-value outputs, for example- it cannot be used for multi-class classification problems. There are steep shifts from 0 to 1, which may not fit the data well. The network is not differentiable, so gradient-based training is impossible.

Sigmoid : The next activation function that we are going to look at is the Sigmoid function. The Sigmoid function performs the role of an activation function in machine learning which is used to add non-linearity in a machine learning model. Basically, the function determines which value to pass as output and what not to pass as output. A sigmoid unit in a neural network. Both the sigmoid and Tanh functions can make the model more susceptible to problems during training, via the so-called vanishing gradients problem. It's one of the best Normalized functions out there. With 1 and 0, it makes a clear prediction.

The sigmoid activation function is defined by the mathematical expression:

$\sigma(z) = 1 / (1 + e^{(-z)})$,

where z represents the input value, and $\sigma(z)$ is the output value between 0 and 1. This function maps any real number to a probability value between 0 and 1.

The two major problems with sigmoid activation functions are: Sigmoid saturate and kill gradients: The output of sigmoid saturates for a large positive or large negative number. Thus, the gradient at these regions is almost zero. One of the major disadvantages of using sigmoid is the problem of vanishing gradient.

Tanh : The tanh function is very similar to the sigmoid function. The only difference is that it is symmetric around the origin. The range of values in this case is from -1 to 1. We observe that the gradient of tanh is four times greater than the gradient of the sigmoid function. This means that using the tanh activation function results in higher values of gradient during training and higher updates in the weights of the network.

The mathematical expression of the Tanh activation function is:

$f(x) = (e^x - e^{-x}) / (e^x + e^{-x})$.

The gradient of the tanh function is steeper as compared to the sigmoid function. Usually tanh is preferred over the sigmoid function since it is zero centered and the gradients are not restricted to move in a certain direction. Tanh tend towards high prediction accuracy. But, a tanh activated neuron may lead to saturation and cause vanishing gradient problem. Issues with tanh activation function: Saturated tanh neuron causes the gradient to vanish. Tanh activation function is computationally expensive due to its exponential operation. Deep neural networks may find it challenging to understand complex associations because Tanh is susceptible to the vanishing gradient problem.

ReLU : The rectified linear activation function is another non-linear activation function that has gained popularity in the deep learning domain. ReLU stands for Rectified Linear Unit. 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. The main advantage of using the ReLU function over other activation functions is that it does not activate all the neurons at the same time. This means that the neurons will only be deactivated if the output of the linear transformation is less than 0.

Mathematically, the ReLU function can be expressed as:

$f(x) = \max(0, x)$ [where x is the input to the function and f(x) is the output]

The drawback of ReLU is that they cannot learn on examples for which their activation is zero. It usually happens if you initialize the entire neural network with zero and place ReLU on the hidden layers. It suffers from a problem known as the dying ReLU which happen during training, some neurons effectively die, meaning they stop outputting anything other than 0. A neuron dies when its weights get tweaked in such a way that the weighted sum of its inputs is negative for all instances in the training set.

ELU : Exponential Linear Unit is a popular machine learning activation function that speeds up training and increases model performance. Exponential Linear Unit is a popular activation function that speeds up learning and produces more accurate results. ELU is also a variant of Rectified

Linear Unit that modifies the slope of the negative part of the function. It is computationally expensive to compute and may be more difficult to implement in certain hardware or software environments. ELU can saturate for large negative inputs, resulting in tiny derivatives and unstable gradients. Unlike the leaky ReLU and parametric ReLU functions, ELU uses a log curve instead of a straight line for defining the negative values. It is leads to faster training times as compared to other linear non-saturating activation functions such as ReLU and its variants. Unlike ReLU, it does not suffer from the problem of dying neurons. This because of the fact that the gradient of ELU is non-zero for all negative values. ELU is slower to compute in comparison to ReLU and its variants because of the non-linearity involved for the negative inputs. During the test time, ELU will perform slower than ReLU and its variants.

SELU : Scaled Exponential Linear Units, are activation functions that induce self-normalization. SELU network neuronal activations automatically converge to a zero mean and unit variance. The SELU activation function is mathematically expressed as a piecewise function that returns lambda times the input if the input is positive, and lambda times a scaled exponential function minus alpha if the input is negative. SELU does not have vanishing gradient problem and hence, is used in deep neural networks. SELU learn faster and better than other activation functions without needing further procession. However, SELU is a relatively new activation function so it is not yet used widely in practice. ReLU stays as the preferred option as compared to SELU. More research on architectures are needed to using SELU for wide-spread industry use.

Activation functions play a key role in neural networks, so it is essential to understand the advantages and disadvantages to achieve better performance. Of course, I won't say that people should use this or that. Because I have listed the unique advantages and disadvantages of each activation function. The sigmoid function can be used if you say that the hyperbolic tangent or model can be learned a little slower because of its wide range of activating functions. But if your network is too deep and the computational load is a major problem, ReLU can be preferred. You can decide to use ELU as a solution to the problem of vanishing gradients in ReLU but ELU has a slower test time. SELU enable deep neural networks without vanishing gradients. SELU is faster and better than other activation functions, even if they are combined with batch normalization. The output of a SELU is normalized, which could be called internal normalization, hence the fact that all the outputs are with a mean of zero and standard deviation of one. The main advantage of SELU is that the Vanishing and exploding gradient problem is impossible and since it is a new activation function, it requires more testing before usage.