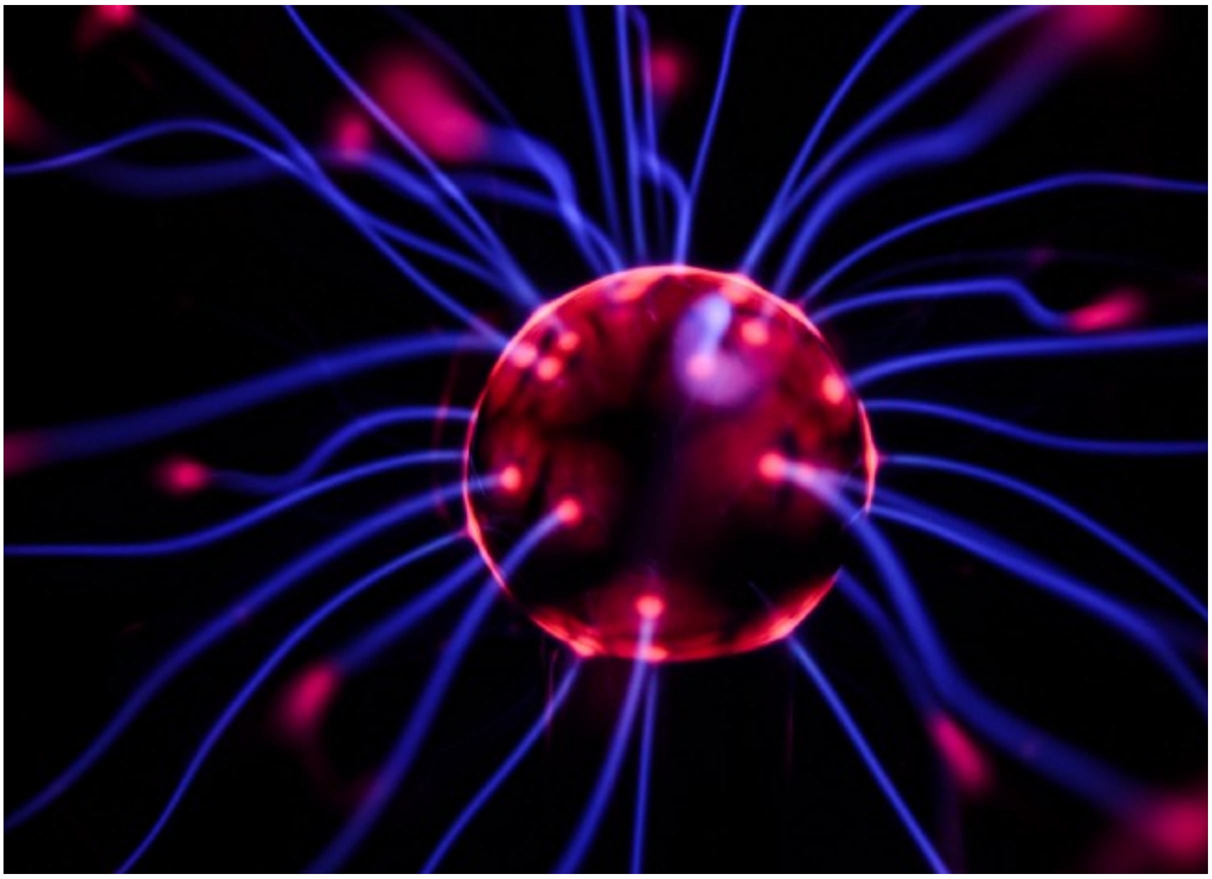
What Are Activation Functions in Deep Learning? - Towards Data Science

David Ben Gurion

Deep Learning / Statistics / TensorFlow



All your questions about activation functions right here in one place. I'll try to keep it short.

What are activation functions?

Why are activation functions required for neural networks?

What are the types of activation function?

How do you implement them in TensorFlow / Keras?

How can you choose the right activation function for you neural network?

Let's start with the most obvious question: '*What are activation functions?*' Assuming all of you deep learning aficionados know the basic structure of an Artificial Neural Network (ANN) and how its building block, the neuron, actually function inside of the network, it must be clear to you that **each neuron**:

1. Receives input(s) from the input layer or neurons from the previous layers in the network
2. Performs some kind of processing or calculation on the information received
3. Sends output signals to the neurons in the next layers of the network

Since neurons perform all of these functions, their outputs vary differently and sometimes are not of much importance to the network. Such information from some neurons when suppressed, will lead to higher scoring networks. This is where activation functions come into play.

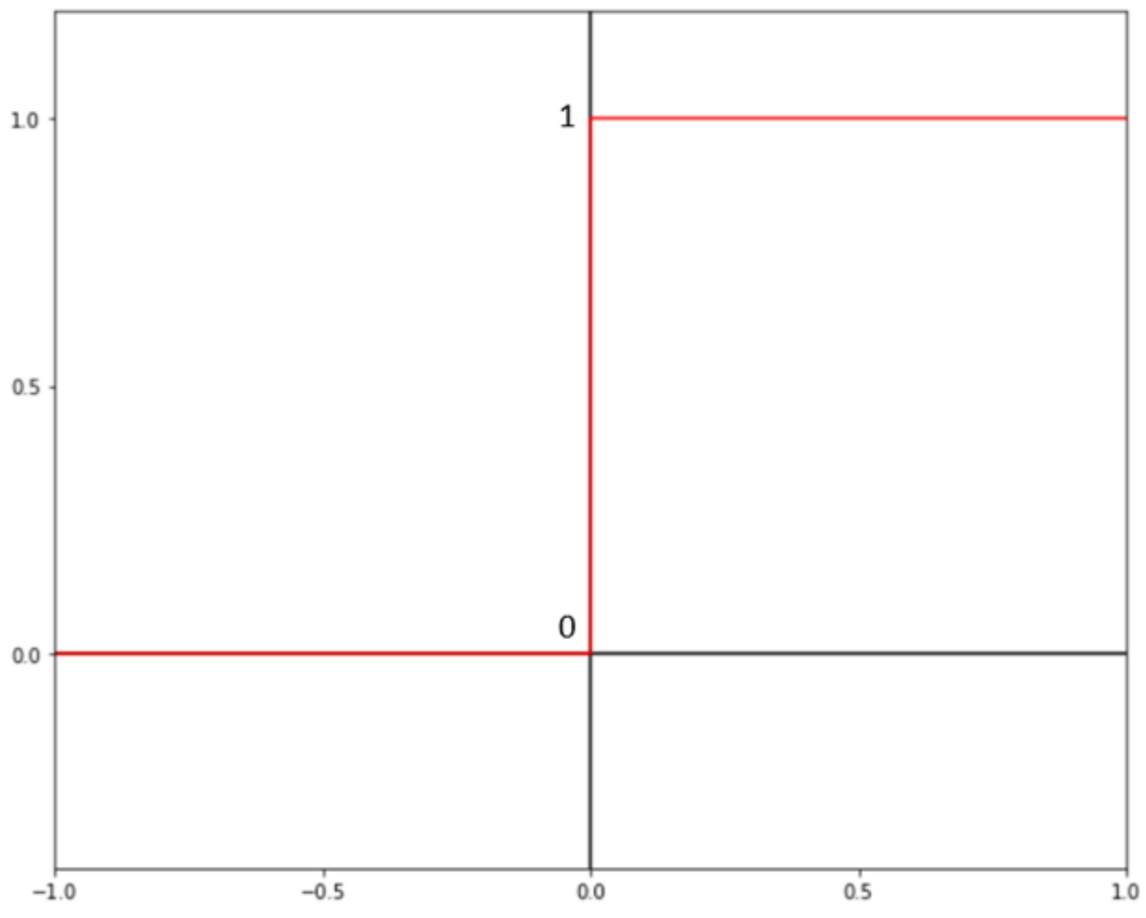
An **activation function** is a deceptively small mathematical expression which decides whether a neuron ***fires up*** or not. This means that the activation function suppresses the neurons whose inputs are of no significance to the overall application of the neural network. This is why neural networks require such functions which provide significant improvement in performance.

Activation functions are in most cases required to provide some kind of non-linearity to the network. Without these functions, the neural network basically becomes a simple linear regression model. However, linear activation functions also exist as we will see below:

Types of Activation Functions:

- Binary Step function
- Linear Activation function
- Non-linear Activation function**s** (quite a few)

1. Binary Step function



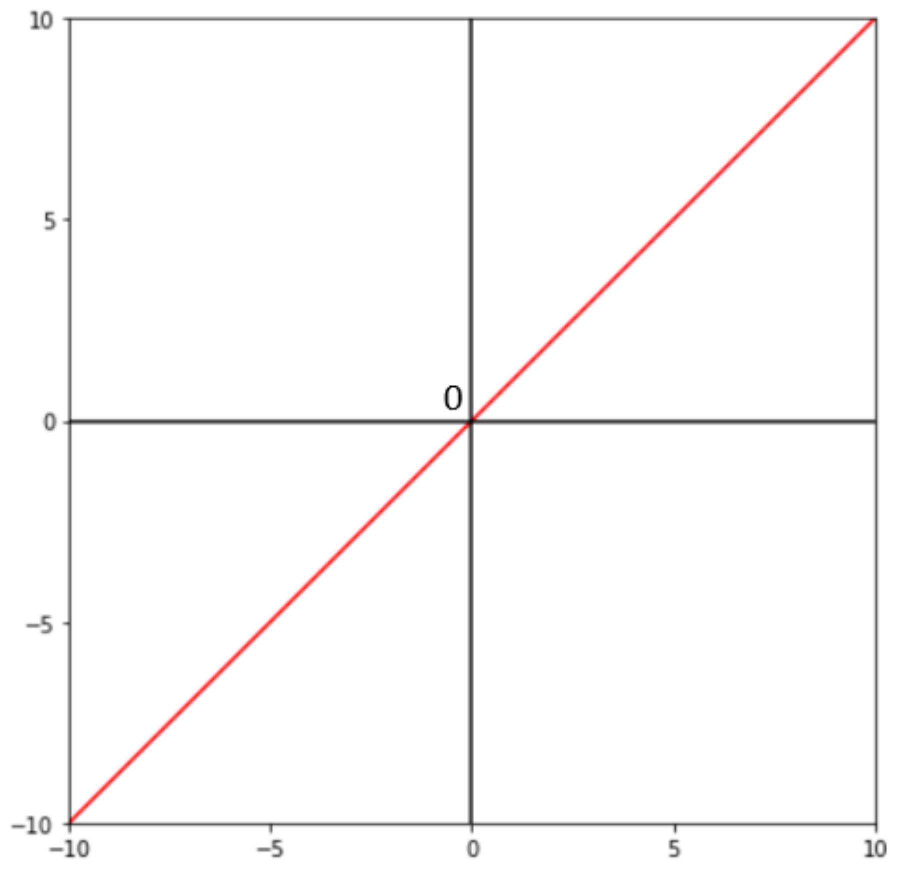
Binary Step Function (Image by Author)

Mathematical Equation:

$f(x) = 0$ for $x < 0$ | 1 for $x \geq 0$

This is the most uncommonly used activation function and you might not even have heard of it. The binary step activation function is a simple threshold classifier. If the input taken by the binary step function is greater than the threshold condition, then the neuron will be activated, meaning that the neuron's output will be successfully passed onto the next layer(s). If not, the neuron will not get fired up and becomes essentially useless to the overall neural network.

2. Linear Activation Function



Linear Activation Function (Image by Author)

Mathematical Equation:

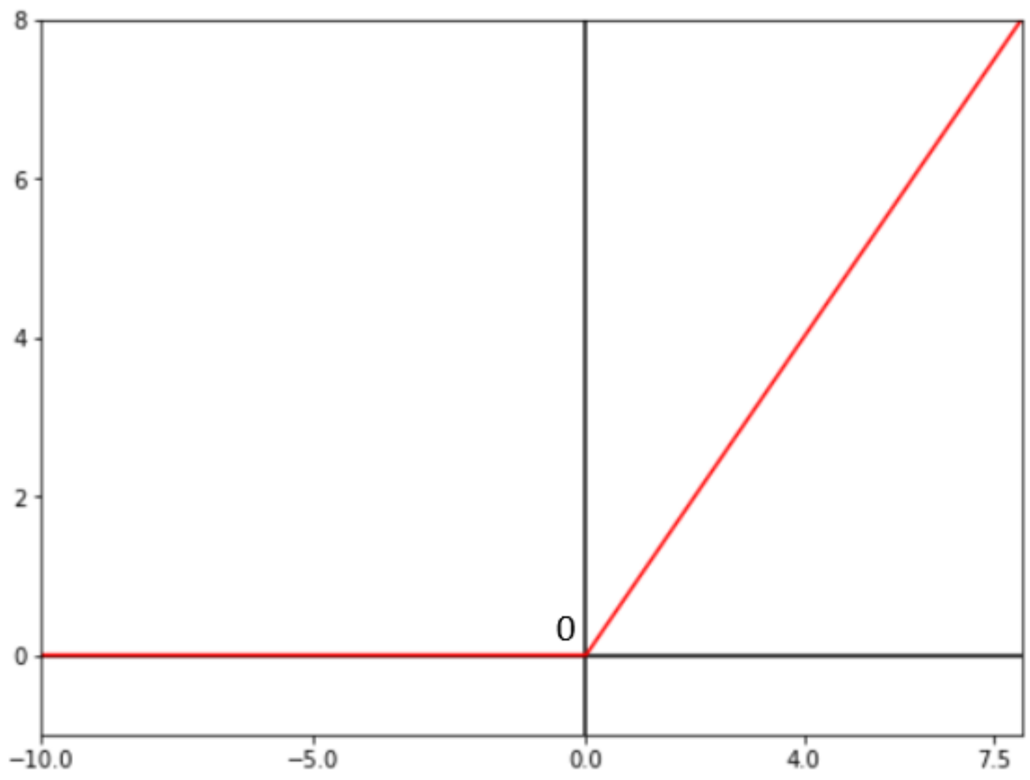
$f(x) = x$

From the equation, it becomes quite obvious that activation is directly proportional to the input received by the function. The gradient here will remain the same for every step of calculation through the network and does not change. For this reason, lienar activation functions should be used for simple regression applications where non-linearity is not required.

3. Non-linear Activation Functions

There’s a bunch of useful non-linear activation functions which can be used for different applications:

• ReLU (Rectified Linear Unit) Activation Function:



ReLU Activation Function (Image by Author)

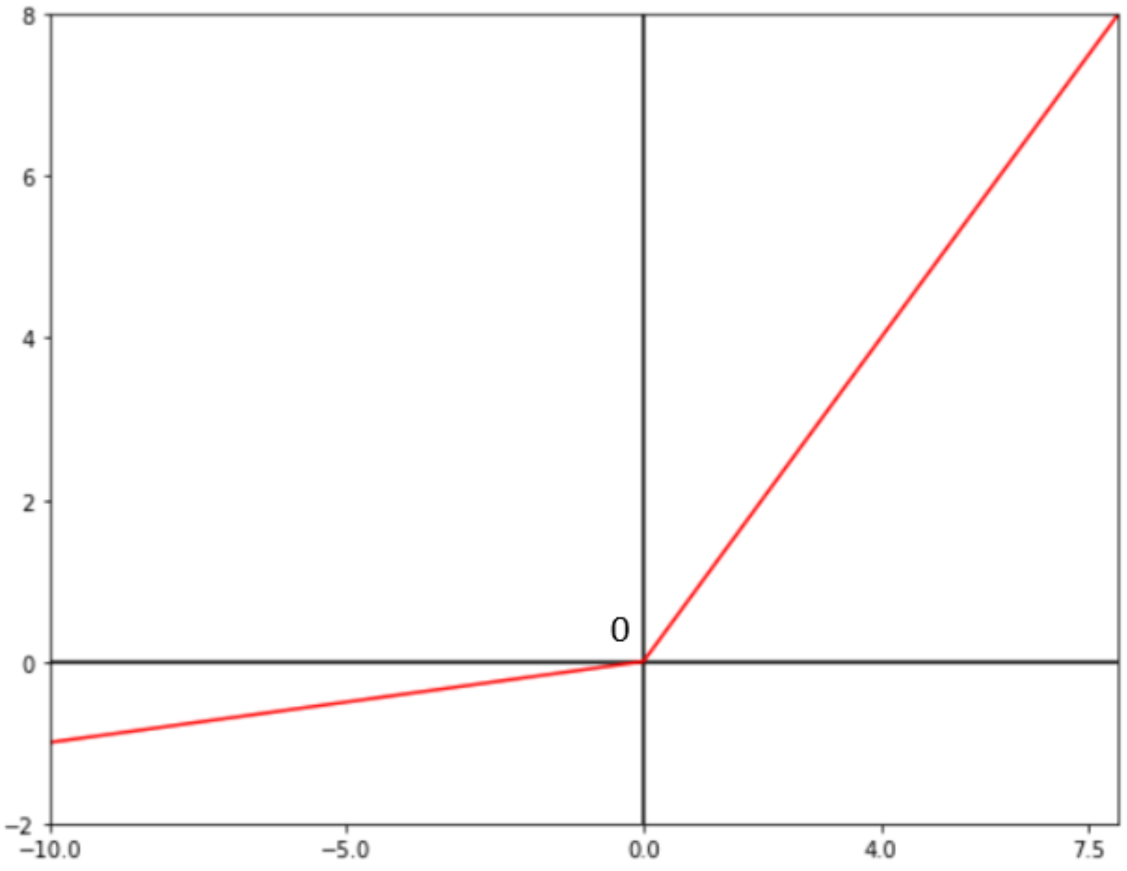
Mathematical Equation:

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

This is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. Hence it is known a the ‘rectified’ linear unit. This is the most common activation function used for the hidden layers in between the input and output layer of a network since it is simple to implement and often results in better performance.

However, since all the negative inputs are mapped as 0, the gradient in this activation becomes zero as well. Due to this, during backpropagation, the weights of some neurons may not be updated rendering them useless. This is known as the **‘Dying ReLU Issue’**.

• Leaky ReLU Activation Function:



Leaky ReLU Activation Function (Image by Author)

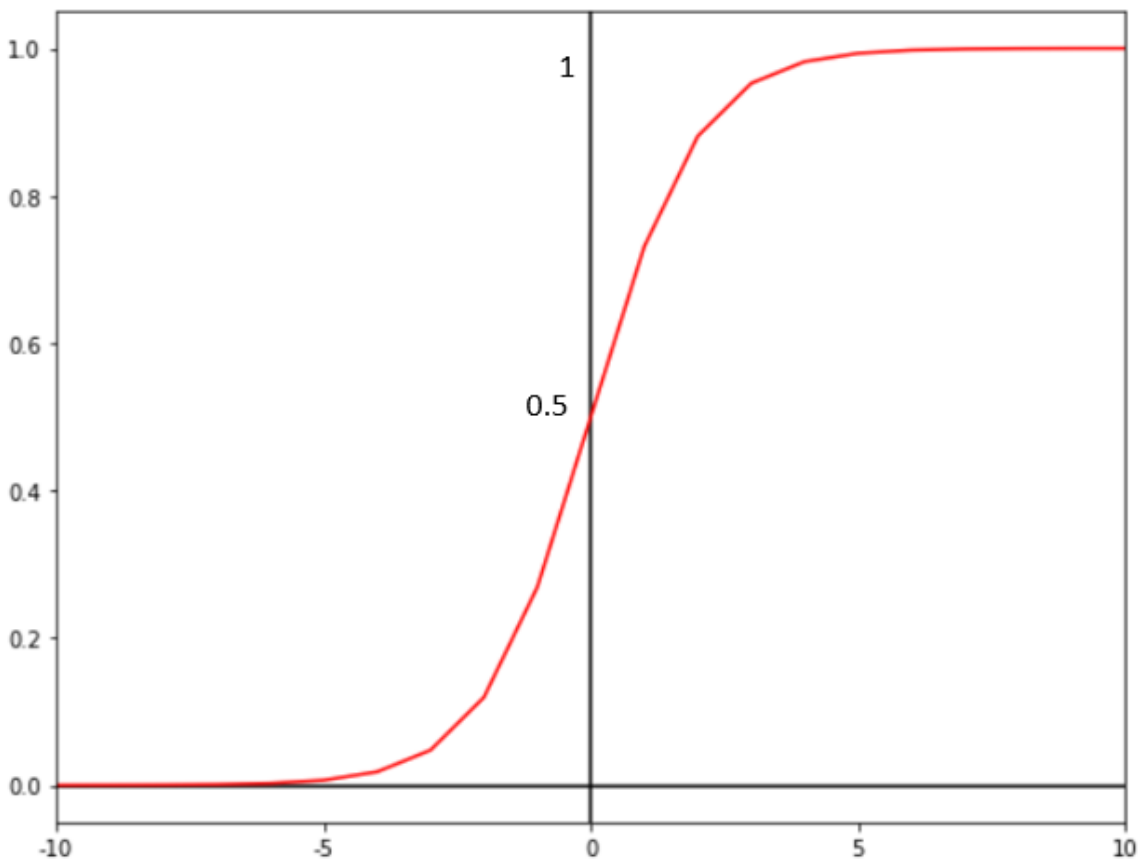
Mathematical Equation:

f(x)=1(x<0)(αx)+1(x>=0)(x); α is a small constant

The Leaky ReLU activation function tries to mitigate the *Dying ReLU problem* which occurs in ReLU with a small positive slope which does not lead to a zero gradient issue. Thus, during backpropagation, the neurons in the negative region are also taken into consideration.

Other rarely used activation functions based on ReLU include: 1) The **Parametric Exponential Linear Unit**(PELU). 2) The **Exponential Linear Unit**(ELU). 3) The **Gaussian Error Linear Unit**(GELU). 4) The **Scaled Exponential Linear Unit**(SELU). They aren't explored here as they are almost never used in most common neural network model and this article aims on being as short as possible.

• Sigmoid Activation Function:



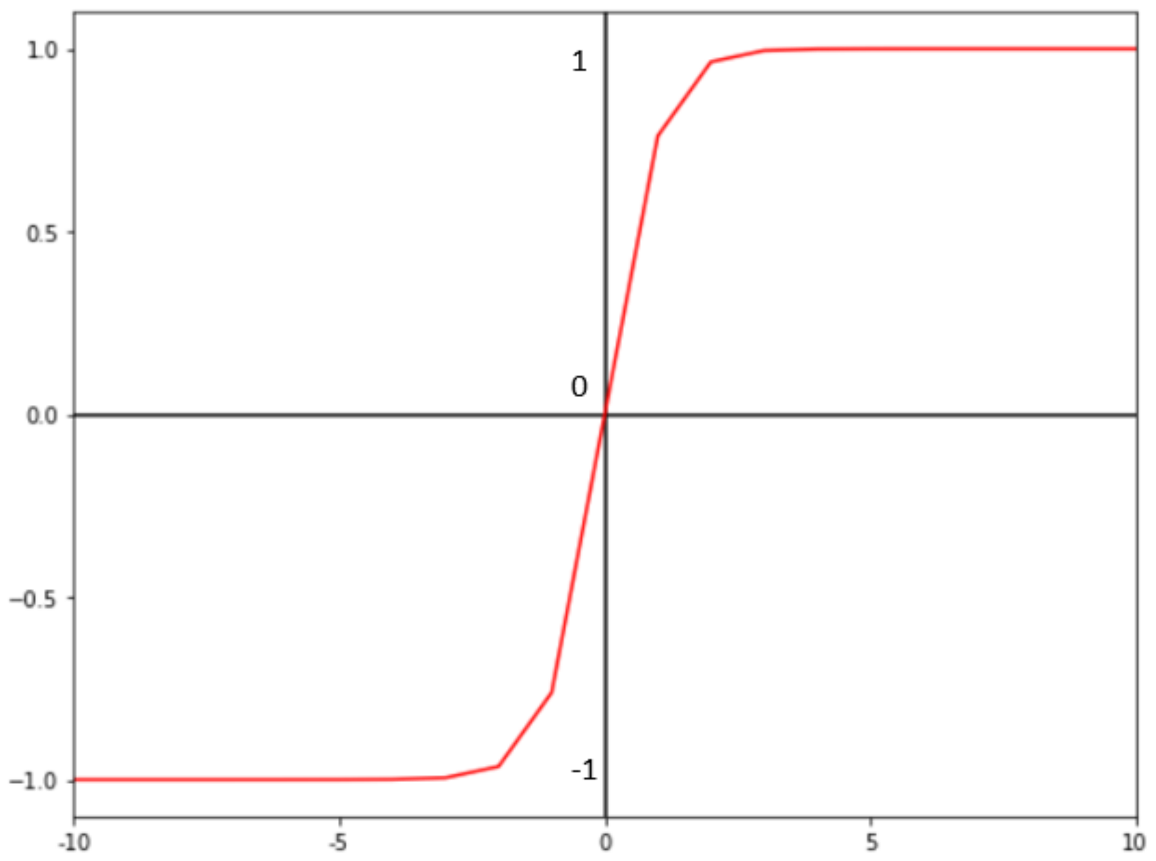
Sigmoid Activation Function (Image by Author)

Mathematical Equation:

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

This function takes in either negative or postive inputs and always returns outputs in the range **(0, 1)**. The largest input is mapped very close to 1 while the smallest input is mapped very close to 0. This is commonly used for application where the probability of a class is required as an output. This is the widely used activation function for the output layer of a **Binary Classification** neural network.

• Tanh Activation Function:



Tanh Activation Function (Image by Author)

Mathematical Equation:

f(x) = (e^x — e^-x) / (e^x + e^-x)

The tanh activation function follows the same gradient curve as the sigmoid function however here, the function outputs results in the range **(-1, 1)**. Because of that range, since the function is zero-centered, it is mostly used in the hidden layers of a neural network. The output from this function can also be negative. Therefore the results can be strongly mapped to the next input neurons.

• Softmax Activation Function:

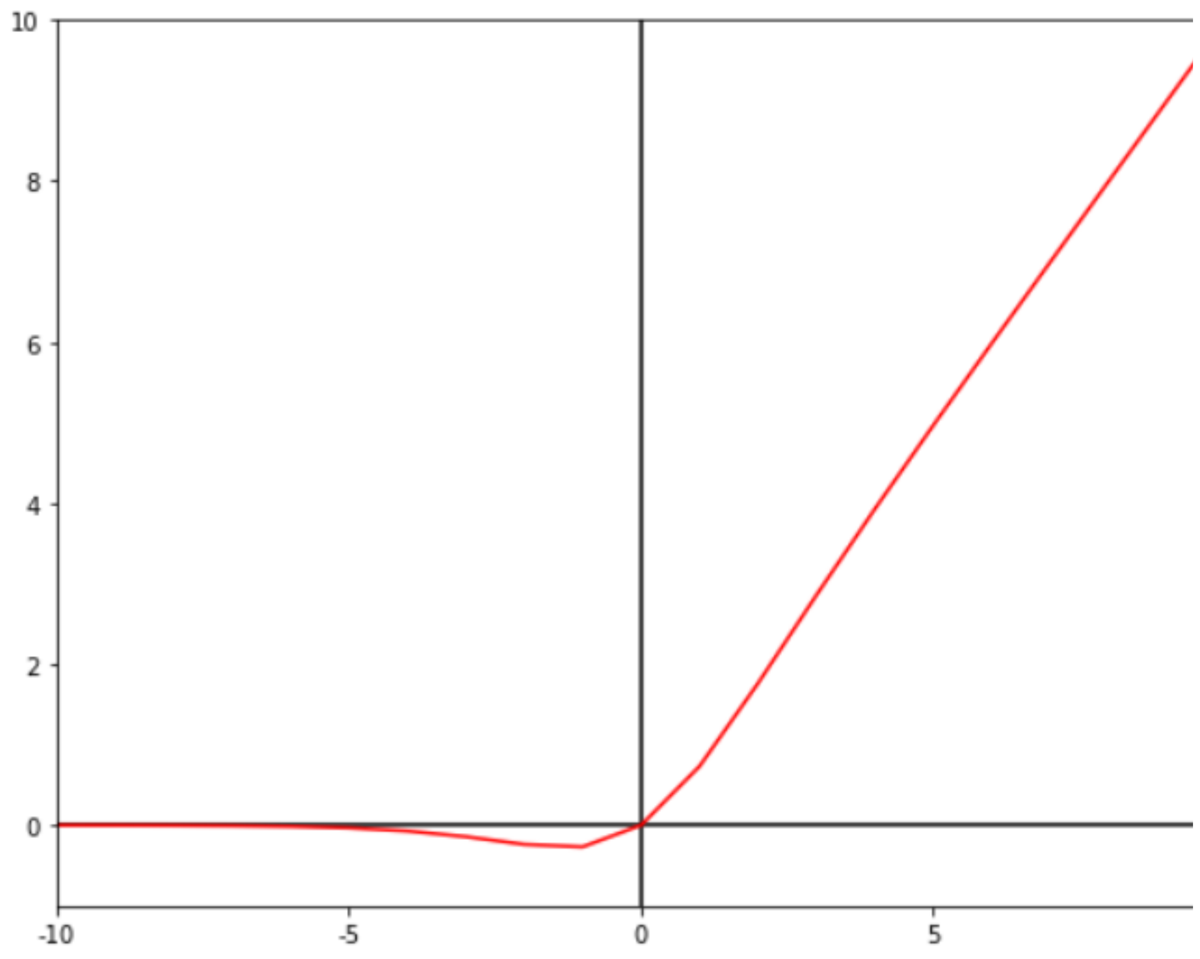
The softmax activation function is based off the sigmoid function which yields the ouputs as a **probability** in the range (0, 1). The softmax function can be defined as a simple combination of mutliple sigmoid functions.

Mathematical Equation:

f(x) = exp(x)/(Σk exp(y)); k is the number of classes, y is the output vector

While the sigmoid function is used for binary classification and yields an output probability, the softmax function is always used for **multi-class classification** applications since it results in the output probabilities of each of the 'k' number of classes in the application which all adds upto 1.

• Swish Activation Function:



Swish Activation Function (Image by Author)

Mathematical Equation:

$f(x) = x \cdot \text{sigmoid}(\beta x)$; β is a learnable parameter

Swish is a self-gated activation function which was only recently proposed by Google. Swish is smoothed out and doesn’t change its direction as abruptly as the ReLU activation function. It can be considered as a modified Leaky-ReLU activation function. Small negative values are not completely suppressed out and thus small underlying patterns from related neurons can also be captured by the network while also evening out the larger negative values to zero which is beneficial for maintaining sparse feature vectors. For this reason, the swish activation function has reportedly outperformed the traditional ReLU function over many different applications by a considerable margin.

Choosing the Right Activation Function for your model:

Here are some handy common rules and conventions that you should follow while building neural networks for different applications.

- **ReLU** activation function is currently the most commonly used function for the **hidden** layers (and never for the output layer) for any type of neural network.
- Although the **swish** activation function does seem to outperform the ReLU function on complex applications, it should be mostly used for only larger neural networks having depths of greater than 50 layers.
- For **binary classification** applications, the output (top-most) layer should be activated by the **sigmoid** function — also for **multi-label classification.**
- For **multi-class** applications, the output layer must be activated by the **softmax** activation function.
- The **linear** activation function should only be used in the output layer of a simple regression neural network.
- For **recurrent neural networks** (RNNs) the **tanh** activation function is preferred for the **hidden layer**(s). It is set by default in **TensorFlow.**
- If the **ReLU** function does not seem to provide the best results, changing the activation to **leaky ReLU** might in some cases yield better results and overall performance.

For Tensorflow activation function implementations, check out this handy module from Tensorflow’s official website.

Thank you for reading. If you liked this article, consider following and clapping.