*International Journal of Engineering Applied Sciences and Technology, 2020*
**Vol. 4, Issue 12, ISSN No. 2455-2143, Pages 310-316**
Published Online April 2020 in IJEAST (http://www.ijeast.com)

# ACTIVATION FUNCTIONS IN NEURAL NETWORKS

Siddharth Sharma, Simone Sharma
UG Scholar, Dept. of Computer Science and Engineering, Global Institute of Technology, Jaipur

Anidhya Athaiya
Assistant Professor, Dept. of Computer Science and Engineering, Global Institute of Technology, Jaipur

**Abstract—Artificial Neural Networks are inspired from the human brain and the network of neurons present in the brain. The information is processed and passed on from one neuron to another through neuro synaptic junctions. Similarly, in artificial neural networks there are different layers of cells arranged and connected to each other. The output/information from the inner layers of the neural network are passed on to the next layers and finally to the outermost layer which gives the output. The input to the outer layer is provided non-linearity to inner layers' output so that it can be further processed. In an Artificial Neural Network, activation functions are very important as they help in learning and making sense of non-linear and complicated mappings between the inputs and corresponding outputs.**

## I. INTRODUCTION

Activation Functions are specially used in artificial neural networks to transform an input signal into an output signal which in turn is fed as input to the next layer in the stack. In an artificial neural network, we calculate the sum of products of inputs and their corresponding weights and finally apply an activation function to it to get the output of that particular layer and supply it as the input to the next layer.

A Neural Network's prediction accuracy is dependent on the number of layers used and more importantly on the type of activation function used. There is no manual that specify the minimum or maximum number of layers to be used for better results and accuracy of the neural networks but a thumb rule shows that a minimum 2 layers to be used. Neither is there any mention in literature of the type of activation function to be used. It is evident from studies and research that using a single/multiple hidden layer in a neural network reduces the error in predictions.

A neural network's prediction accuracy is defined by the type of activation function used. The most commonly used activation functions are non-linear activation functions. A neural network works just like a linear regression model where the predicted output is same as the provided input if an activation function is not defined. Same is the case if a linear activation function is used where the output is similar as the input fed along with some error. A linear activation function's boundary is linear and the if they are used, then the network can adapt to only the linear changes of the input
but, in real world the errors possess non-linear characteristics which in turn with the neural networks ability to learn about erroneous data. Hence non-linear activation functions are preferred over linear activation functions in a Neural Network.

The most appealing property of Artificial Neural Networks is the ability to adapt their behavior according to the changing characteristics of the system. In the last few decades many researchers and scientists have performed studies and investigated a number of methods to improve the performance of Artificial Neural Networks by optimizing the training methods, hyperparameter tuning, learn parameters or network structures but not much attention has been paid towards activation functions.

## II. NEURAL NETWORKS

According to inventor of one of the first neurocomputer, neural network can be defined as:
"...a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs.

Artificial Neural Networks are based on the network of neurons in the mammalian cortex and are modelled loosely but on a much smaller scale. Artificial Neural Networks can be algorithms or an actual piece of hardware. There are billions of neurons present in the mammalian brain which gives enormous magnitude of the interaction and emergent behavior, but in an Artificial Neural Network there may have hundreds or thousands of processor units which is very small as compared to mammalian brain structure.

Neural Networks are organized in multiple layers and each layer is made up of a number of interconnected nodes which have activation functions associated with them. Data is fed to the network via the input layer which then communicates with other layers and process the input data with the help of a system of weighted connections. This processed data is then obtained through the output layer.

## III. WHY NEURAL NETWORKS NEED ACTIVATION FUNCTIONS?

Neural Networks are a network of multiple layers of neurons consisting of nodes which are used for classification and prediction of data provided some data as input to the network. There is an input layer, one or many hidden layers and an output layer. All the layers have nodes and each node has a weight which is considered while processing information from one layer to the next layer.

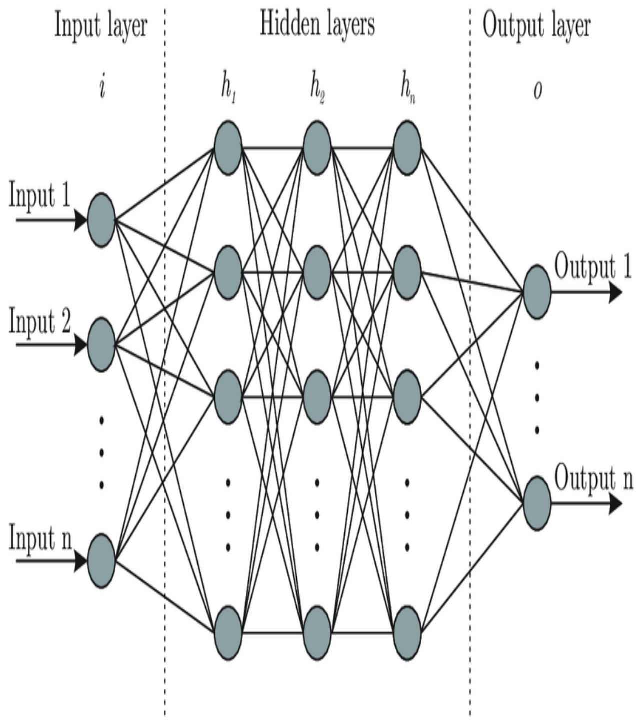

***Figure 1. Neural Network***

If an activation function is not used in a neural network then the output signal would simply be a simple linear function which is just a polynomial of degree one. Although a linear equation is simple and easy to solve but their complexity is limited and they do not have the ability to learn and recognize complex mappings from data. Neural Network without an activation functions acts as a Linear Regression Model with limited performance and power most of the times. It is desirable that a neural network not just only learn and compute a linear function but perform tasks more complicated than that like modelling complicated types od data such as images, videos, audio, speech, text, etc.

This is the reason that we use activation functions and artificial neural network techniques like Deep Learning that makes sense of complicated, high dimensional and non-linear datasets where the model has multiple hidden layers and also a complex architecture for extracting knowledge, which again is our ultimate goal.

## IV. THE NEED FOR NON-LINEARITY IN NEURAL NETWORKS

Those functions which have degree more than one and have a curvature when plotted are known as Non-linear functions. It is required of a neural network to learn, represent and process any data and any arbitrary complex function which maps the inputs to the outputs. Neural Networks are also known as Universal Function Approximators which, means that they can compute and learn any function provided to them. Any imaginable process can be represented as a functional computation in Neural Networks.

Thus, we need to apply an activation function to make the network dynamic and add the ability to it to extract complex and complicated information from data and represent non-linear convoluted random functional mappings between input and output. Hence, by adding non linearity with the help of non-linear activation functions to the network, we are able to achieve non-linear mappings from inputs to outputs. An important feature of an activation function is that it must be differentiable so that we can implement back propagation optimization strategy in order to compute the errors or losses with respect to weights and eventually optimize weights using Gradient Descend or any other optimization technique to reduce errors.

## V. TYPES OF ACTIVATION FUNCTIONS

Net inputs are the most important units in the structure of a neural network which are processed and changed into an output result known as unit’s activation by applying function called the activation function or threshold function or transfer function which is a a scalar to scalar transformation.

To enable a limited amplitude of the output of a neuron and enabling it in a limited range is known as squashing functions. A squashing function squashes the amplitude of output signal into a finite value.

1. Binary Step Function
2. Linear
3. Sigmoid
4. Tanh
5. ReLU
6. Leaky ReLU
7. Parametrized ReLU
8. Exponential Linear Unit
9. Swish
10. SoftMax

### a. BINARY STEP FUNCTION

When we have an activation function, the most important thing to consideration is threshold based classifier which means that whether the linear transformation's value must activate the neuron or not or we can say that a neuron gets activated if the input to the activation function is greater than a threshold value, or else it gets deactivated. In that case the output is not fed as input to the next layer.

Binary Step Function is the simplest activation function that exists and it can be implemented with simple if-else statements in Python. While creating a binary classifier binary activation function are generally used. But, binary step function cannot be used in case of multiclass classification in target carriable. Also, the gradient of the binary step function is zero which may cause a hinderance in back propagation step i.e if we calculate the derivative of f(x) with respect to x, it is equal to zero. Mathematically binary step function can be defined as:

f(x) = 1, x>=0
f(x)= 0, x<0

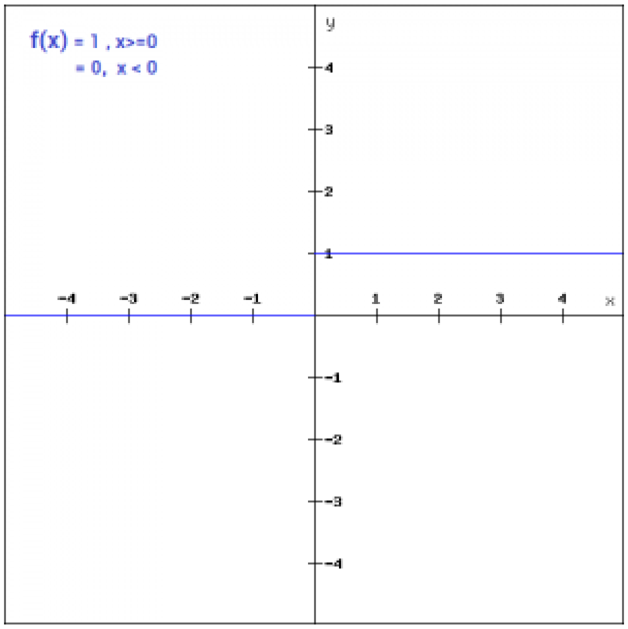

*Figure 2. Binary Step Function*

b. LINEAR ACTIVATION FUNCTION

The linear activation function is directly proportional to the input. The main drawback of the binary step function was that it had zero gradient because there is no component of x in binary step function. In order to remove that, linear function can be used. It ca be defined as:

F(x) = ax

The value of variable a can be any constant value chosen by the user.

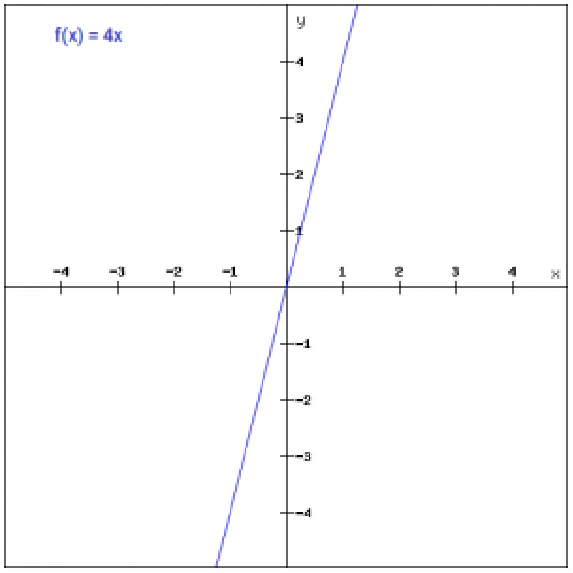

*Figure 3. Linear activation function*

Here the derivative of the function f(x) is not zero but is equal to the value of constant used. The gradient is not zero, but a constant value which is independent of the input value x which implies that the weights and biases will be updated during backpropagation step although the updating factor will be the same.

There isn't much benefit of using linear function because the neural network would not improve the error due to the same value of gradient for every iteration. Also, the network will not be able to identify complex patterns from the data. Therefore, linear functions are ideal where interpretability is required and for simple tasks.

c. SIGMOID ACTIVATION FUNCTION

It is the most widely used activation function as it is a non-linear function. Sigmoid function transforms the values in the range 0 to 1. It can be defined as:

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

Sigmoid function is continuously differentiable and a smooth S-shaped function. The derivative of the function is:

f'(x) = 1-sigmoid(x)

Also, sigmoid function is not symmetric about zero which means that the signs of all output values of neurons will be same. This issue can be improved by scaling the sigmoid function.

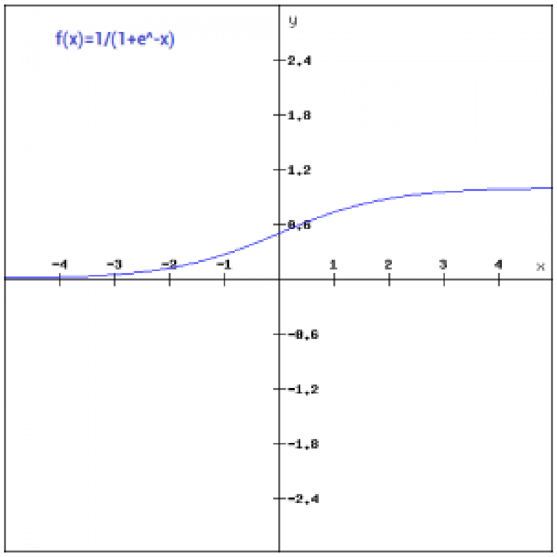

*Figure 4. Sigmoid Function*

d. TANH FUNCTION

It is Hyperbolic Tangent function. Tanh function is similar to the sigmoid function but it is symmetric to around the origin. This results in different signs of outputs from previous layers which will be fed as input to the next layer. It can be defined as:

f(x) = 2sigmoid(2x)-1

Tanh function is continuous and differentiable, the values lies in the range -1 to 1. As compared to the sigmoid function the gradient of tanh function is more steep. Tanh is preferred over sigmoid function as it has gradients which are not restricted to vary in a certain direction and also, it is zero centered.

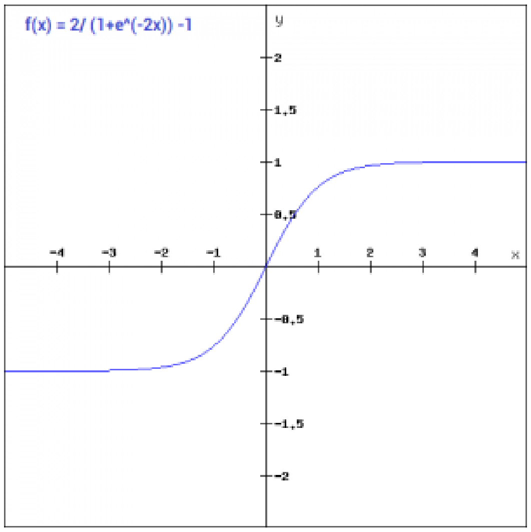

*Figure 5. Tanh Function*

e. RELU FUNCTION

ReLU stands for rectified liner unit and is a non-linear activation function which is widely used in neural network. The upper hand of using ReLU function is that all the neurons are not activated at the same time. This implies that a neuron will be deactivated only when the output of linear transformation is zero. It can be defuned mathematically as:

f(x) = max(0,x)

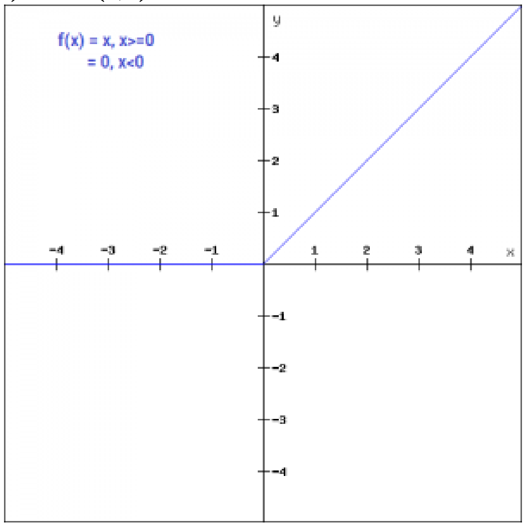

*Figure 6. ReLU Activation Function plot*

ReLU is more efficient than other functions because as all the neurons are not activated at the same time, rather a certain number of neurons are activated at a time.

In some cases, the value of gradient is zero, due to which the weights and biases are not updated during back-propagation step in neural network training.

f. LEAKY RELU FUNCTION

Leaky ReLU is an improvised version of ReLU function where for negative values of x, instead of defining the ReLU functions' value as zero, it is defined as extremely small linear component of x. It can be expressed mathematically as:

f(x) = 0.01x, x < 0

f(x) = x, x >= 0

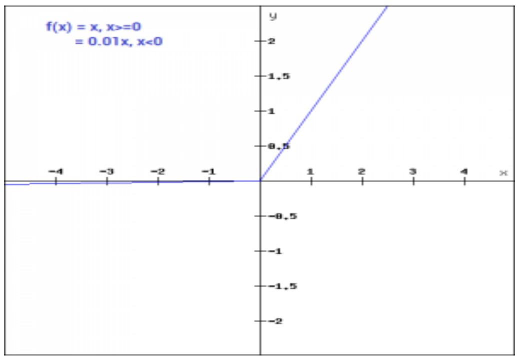

*Figure 7. Plot of Leaky ReLU functiom*

g. PARAMETRIZED RELU FUNCTION

It is also a variant of Rectified Linear Unit with better performance and a slight variation. It resolves the problem of gradient of ReLU becoming zero for negative values of x by introducing a new parameter of the negative part of the function i.e Slope.
It is expressed as:
f(x) = x, x >= 0
f(x) = ax, x < 0

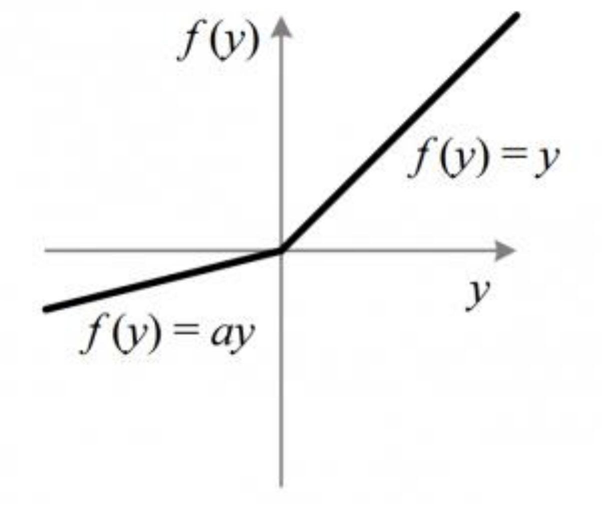

*Figure 8. Plot of Parametrized ReLU function*

The value of a, when set to 0.01, it behaves as leaky ReLU function but here a is also a trainable parameter. For faster and optimum convergence, the network learns the value of a.

h. EXPONENTIAL LINEAR UNIT

Exponential Linear Unit or ELU is also a variant of Rectified Linear Unit. ELU introduces a parameter slope for the negative values of x. It uses a log curve for defining the negative values.
f(x) = x, x >= 0
f(x) = a($e^x$ – 1), x < 0

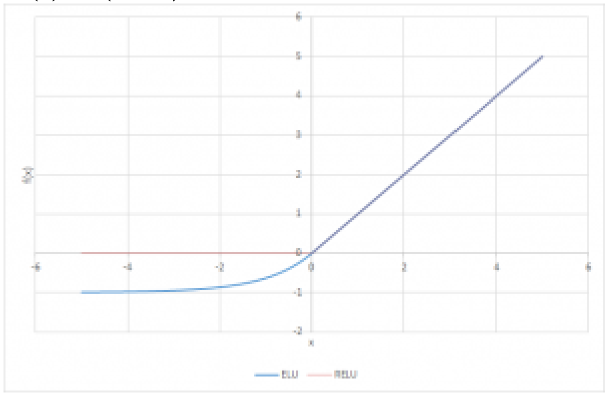

*Figure 9. Plot of ELU function*

i. SWISH FUNCTION

Swish function is a relatively new activation function which was discovered by researchers at GOOGLE. The distinguishing feature of Swish function is that it is nit Monotonic, which means that the value of function may decrease even though the values of inputs are increasing. In some cases, Swish outperforms even the ReLU function.
It is expressed mathematically as:
f(x) = x * sigmoid(x)
f(x) = x/(1 – $e^{-x}$)

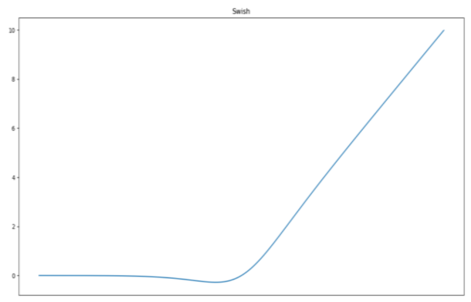

*Figure 10. Swish Function*

j. SOFTMAX ACTIVATION FUNCTION

Softmax function is a combination of multiple sigmoid functions. As we know that a sigmoid function returns values in the range 0 to 1, these can be treated as probabilities of a particular class' data points.

Softmax function unlike sigmoid functions which are used for binary classification, can be used for multiclass classification problems. The function, for every data point of all the individual classes, returns the probability. It can be expressed as:

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

When we build a network or model for multiple class classification, then the output layer of the network will have the same number of neurons as the number of classes in the target.

## VI. CHOOSING THE RIGHT ACTIVATION FUNCTION

For better performance and less erroneous results, a lot of things have to be considered like the number of hidden layers in a network, training methods, hyperparameter tuning etc and activation function is one of the most important parameters to consider. Choosing the right activation function for any particular task may be a tedious process and may require a lot of research and studies.

There is no thumb rule for selecting any activation function but the choice of activation function is context dependent, i.e it depends on the task that is to be accomplished. Different Activation Functions have both advantages and dis-advantages of their own and it depends on the type of system that we are designing. For example.:

- For classification problems, a combination of sigmoid functions gives better results.
- Due to vanishing gradient problem i.e. gradient reaching the value zero, sigmoid and tanh functions are avoided.
- ReLU function is the most widely used function and performs better than other activation functions in most of the cases.
- If there are dead neurons in our network, then we can use the leaky ReLU function.
- ReLU function has to be used only in the hidden layers and not in the outer layer.

We can experiment with different activation functions while developing a model if time constraints are not there. We start with ReLU function and then move on to other functions if it does not give satisfactory results.

As studies have shown that both sigmoid and tanh functions are not suitable for hidden layers because the slope of function becomes very small as the input becomes very large or very small which in turn slows down gradient descent. ReLu is the most preferred choice for apllying with hidden layers as the derivative of ReLU is 1. Also, leaky ReLU can be used in case of zero derivatives. An activation function which is going to approximate the function faster and can be trained at a faster rate can also be chosen.

## VII. CONCLUSION

This paper provides a brief description of various activation functions that are used in the field of deep learning and also about the importance of activation functions in developing an effective and efficient deep learning model and improving the performance of artificial neural networks.
This paper highlights the need of activation function and the need for non-linearity in neural networks. Firstly, we have given a description of activation functions, then we have given a brief description of the need of activation functions and the need of non-linearity in neural networks. We then describe various types of activation functions that are commonly used in neural networks. Activation Functions have the ability to improve the learning rate and learning of patterns present in the dataset which, in turn, helps in automation of process of feature detection, extraction and predictions. This paper justifies the use of activation functions in the hidden layers of neural networks and their usefulness in classification across various domains. These activation functions were developed after extensive research and experiments over the years.

There has been little emphasis on activation functions in the past but now, at present scientists and developers are paying the much-needed attention and research in the study of activation functions as they affect neural network's performance. Also, there are various activation functions that are not discussed in this literature as they aren't widely used in deep learning but rather we have emphasized on the most commonly used activation functions In future we could also work to compare all the activation functions and analyze their performance using standard datasets and architectures to see if their performance can be further improved.

## VIII. REFERENCES

[1.] Karlik, B., & Olgac, A. V. (2011). Performance analysis of various activation functions in generalized MLP architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, *1*(4), 111-122.

[2.] Agostinelli, F., Hoffman, M., Sadowski, P., & Baldi, P. (2014). Learning activation functions to improve deep neural networks. *arXiv preprint arXiv:1412.6830*.

[3.] Chen, T., & Chen, H. (1995). Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, *6*(4), 911-917.

[4.] Stinchcombe, M., & White, H. (1989, December). Universal approximation using feedforward networks with non-sigmoid hidden layer activation functions. In *IJCNN International Joint Conference on Neural Networks*.

[5.] Cao, J., & Wang, J. (2004). Absolute exponential stability of recurrent neural networks with Lipschitz-continuous activation functions and time delays. *Neural networks*, *17*(3), 379-390.

[6.] Huang, G. B., & Babri, H. A. (1998). Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions. *IEEE transactions on neural networks*, *9*(1), 224-229.

[7.] Sibi, P., Jones, S. A., & Siddarth, P. (2013). Analysis of different activation functions using back propagation neural networks. *Journal of Theoretical and Applied Information Technology*, *47*(3), 1264-1268.

[8.] Ma, L., & Khorasani, K. (2005). Constructive feedforward neural networks using Hermite polynomial activation functions. *IEEE Transactions on Neural Networks*, *16*(4), 821-833.

[9.] Lu, W., & Chen, T. (2005). Dynamical behaviors of Cohen–Grossberg neural networks with discontinuous activation functions. *Neural Networks*, *18*(3), 231-242.

[10.] Gomes, G. S. D. S., Ludermir, T. B., & Lima, L. M. (2011). Comparison of new activation functions in neural network for forecasting financial time series. *Neural Computing and Applications*, *20*(3), 417-439.

[11.] Hashem, S. (1992, June). Sensitivity analysis for feedforward artificial neural networks with differentiable activation functions. In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks* (Vol. 1, pp. 419-424). IEEE.

[12.] Piazza, F., Uncini, A., & Zenobi, M. (1993, October). Neural networks with digital LUT activation functions. In *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)* (Vol. 2, pp. 1401-1404). IEEE.

[13.] Yi, Z., & Tan, K. K. (2004). Multistability of discrete-time recurrent neural networks with unsaturating piecewise linear activation functions. *IEEE Transactions on Neural Networks*, *15*(2), 329-336.