The Most Used Activation Functions: Classic Versus Current

Marina Adriana Mercioni
Department of Computer Science
Politehnica University Timisoara
Timisoara, Romania
marina.mercioni@student.upt.ro

Stefan Holban
Department of Computer Science
Politehnica University Timisoara
Timisoara, Romania
stefan.holban@cs.upt.ro

Abstract—**This paper is an overview of the most used activation functions, classic functions and current functions as well. When we say classic, we mean the first activation functions, the most popular and used in the past. But due to their disadvantages appeared other new activation functions that we refer them as current. These most influential functions are among the most known artificial intelligence activation functions in the research of Machine learning and Deep Learning as well. With each function, we provide a brief description of the activation function, discuss its impact and show the domain where it is applicable, its advantages and disadvantages and more details to have an overview. These functions cover more issues like vanishing gradient, exploding gradient when we are using Gradient Descent and so on. These solutions to these issues are all among the most important topics in Artificial Intelligence research and development part.**

Keywords—activation function; backpropagation; gradient descent; optimization; classic; current; sigmoid; softmax; relu; artificial intelligence; machine learning; deep learning;

I. INTRODUCTION

The Artificial Neural Networks (ANNs) introduced in the 1950's year became the most studied domain in the last years. In 2006 Hinton developed Gradient Descent algorithm but it did not give good results without activation functions. The activation functions represent the continuous development of Artificial Intelligence.

Nowadays more activation functions developed and they permit to use them in different ways which helps the networks to reach faster the convergence or they give to the networks the capability to use fewer layers in network's architecture. With fewer layers in architecture, we have fewer parameters in our network and it consists of a good optimization network.

Artificial Neural Networks can create non-linear functions. So, the main purpose of the activation function is to introduce non-linearity in the outcome of a neuron. Also, the activation function known as transfer function can decide if a neuron must be activated or not through a weighted sum and adding the bias. A neural network without activation functions is only a linear regression model. But the activation function gives the network the capacity to learn more complex tasks.

In the last period the artificial neural networks became very popular, giving the highest performances for a diversity of applications, including **N**atural **L**anguage **P**rocessing (NLP), Computer Vision and so on. [1]

Another important aspect of the activation functions and weights' initialization determination, was given by space where the optimization algorithm will take. It was coming as a solution for the network's creation which consists of a non-convex optimization issue. [2]

How we said earlier the activation functions are the most used by artificial neural networks, so they represent a standard in developing of network architecture's type depending by their purpose. [3] [4]

The selection of an activation function represents an important topic because it can affect the manner how the input data are changed.

Bengio *et al.*, proved that a new type of activation function known as ReLU (**R**ectified **L**inear **U**nit) improves the network's performance from point of time efficiency and the complexity's space. Also, they studied the impact of using non-linear behavior instead of sigmoid function or tangent function(known as tanh) through using of regularizer to prevent possible numerical issues with unbounded activations. In 2010 Hinton showed good results using Restricted Boltzmann Machine instead of the sigmoid function. [5]

II. CLASSIC ACTIVATION FUNCTIONS

A. Sigmoid function

Some remarkable properties of sigmoid function (also known as a logistic function) are: continuous behavior (smoothed), grow monotone, and delimited. The sigmoid function is delimited between 0 and 1. In our study, we used the MNIST handwritten digit classification problem which is a standard dataset used in Computer Vision and Deep Learning.

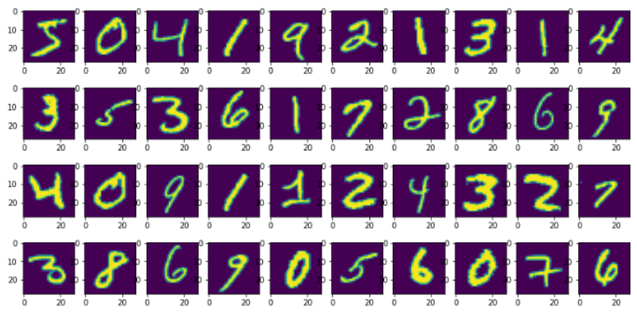


Fig. 1. A subset of images from the MNIST Dataset

It helped us to develop a robust test harness for estimating the performance of our model, how to explore improvements to the model, and how to make predictions on new data. For better visualization we tested both classic and current functions on MNIST dataset (Yann LeCun), we can see their plots in Figure 1 as follows:

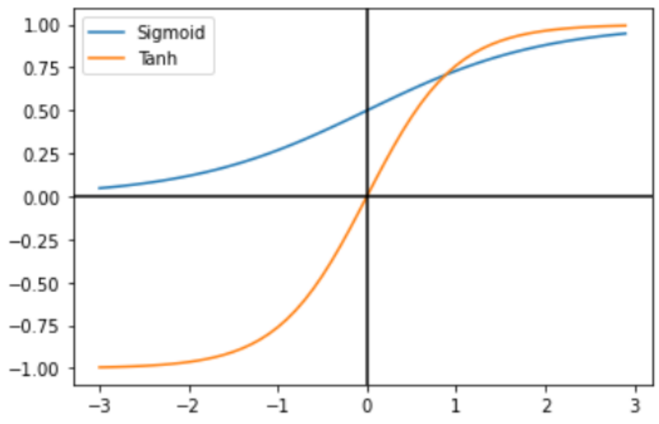


Fig. 2. Sigmoid versus tanh functions on MNIST Dataset using two classic functions

The sigmoid function is a suitable choice when the outcome's function takes a value between 0 and 1. This function is used e.g. for the classification tasks in artificial neural networks. This function was used very intensely in the Machine Learning foundation, most for logistic regression and other few implementations. But also, this function has disadvantages as well.

The sigmoid equation is the following:

$$f(x) = sigmoid(x) = \frac{1}{1+e^{-x}} \tag{1}$$

Starting from this equation we can compute easily the derivative function as:

$$f'(x) = f(x)(1 - f(x)) \tag{2}$$

As we can see above the derivative of a function can be computed easily but this property forces the model to lose data features and drives to a decrease of performance's model.

The main disadvantage of the sigmoid function is related to its usefulness in the backpropagation algorithm. Other important disadvantages of this activation which we can mention:

- Saturated neurons "kill" the gradients (the gradients are very small and become equal to 0).
- The outcome of the sigmoid function is not centered on 0.
- The sigmoid function uses exponential compute so it drives to high computational cost.
- Another minus is the exploding of gradients if we use recurrent architecture as LSTM (**L**ong **S**hort-**T**erm **M**emory) or GRU (**G**ated **R**ecurrent **U**nit).

B. *Tangent function (tanh)*

Tanh function developed by LeCun *et al.*, it was more efficiently in hidden layers. It comes as a solution for the sigmoid minus when the outcomes are negative, and sigmoid produces an outcome close to 0. That means a difficult change of weights associated with that hidden layer, so it becomes a "dead" node. Tanh function has the same properties as sigmoid but it delimited between -1 and 1.

The tanh equation is the following:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3}$$

The main characteristics of tanh function are:

- Brings the numbers in the range [-1, 1].
- The media is equal to 0.
- It "kills" the gradients when it reaches saturation.

III. The Current Functions

With a continuous desire to improve the artificial intelligence domain, more functions appeared in different shapes to bring solutions for the previous issues and they became key functions in different areas where they are used such as **C**onvolutional **N**eural **N**etworks (CNN). A popular function that gives very good results is ReLU (**R**ectified **L**inear **U**nit). In the next subchapters, we will discuss these functions.

A. *ReLU (**R**ectified **L**inear **U**nit)*

It is not smoothed and limited but it works well in the applications with a big number of units such as Convolutional Neural Networks.

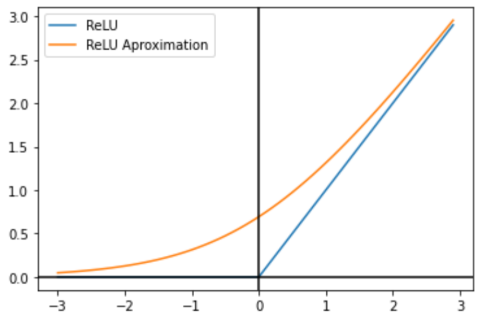


Fig. 3. ReLU approximation and ReLU function on MNIST dataset

ReLU introduced by Krizhevsky *et al.* in 2012, is a linear function that returns the input if the input is positive otherwise it returns 0. In Stochastic Gradient Descent with backpropagation of the errors for train the model, it's necessary a function that to behave as a linear function, but it is a nonlinear function permitting to learn complex relations of data. And ReLU exactly this thing does.

The derivative of ReLU function is easy to compute. Its derivative is the slope. For negative values, this slope is equal to 0 and 1 for positive values.

Some remarkable advantages of ReLU functions are:

- Computational simplify – in 2012, Xavier Glorot *et al*., showed through their paper entitled "*Deep Sparse Rectifier Neural Networks*", that ReLU does not need exponential computation.
- Representative sparsity: This property gives the capacity to have True value 0. That means the negative inputs can return True 0, permitting the activation of hidden layers to contain one or more True 0 values. This property is known as representative sparsity and it is very desired because it accelerates the training and simplifies the model.
- Linear behavior – ReLU looks and acts as a linear activation function. Another plus of this function is the usefulness in the training of deep neural networks.

The ReLU advantages are listed below:

- Less complexity of time and space in comparison with sigmoid, it does not need exponential computation which is more expensive.
- It solves the issue of vanishing gradient.
- It does not saturate on the positive side.
- Very efficiently computationally.
- In practice, it converges faster than sigmoid vs tanh (e.g. 6x).

Some disadvantages we can mention:

- The outcome does not have the media equal to 0.
- An unfavorable case is given by the negative values. (When $x<0$, the gradient is equal to 0).
- ReLU does not avoid the exploding gradient issue.

*B. PReLU (**P**arametric **R**ectified **L**inear **U**nit)*

He *et al*., developed a new activation function called PReLU (**P**arametric **R**ectified **L**inear **U**nit) that is derived from ReLU, it brings improvements to the model with computational needs rough 0 and reduced overfitting risks. [6]

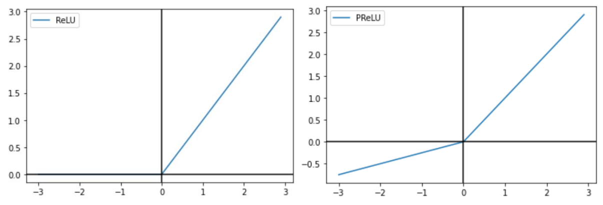


Fig. 4. ReLU vs PReLU functions on MNIST dataset

The experiments on the ImageNet dataset [7] proved that PReLU has a better performance than ReLU, resulting also a smaller loss.

*C. Leaky ReLU (**L**eaky **R**ectified **L**inear **U**nit)*

In [8] the authors solved the minus of ReLU function through introducing the newest shape of ReLU called ***Leaky ReLU*** (LReLU). The leaky parameter permits obtaining a small non-zero gradient when the unit is saturated and it is not active.
The equation of PReLU is given by:

$$f(x) = \max(0.01x, x) \tag{4}$$

The biggest advantage of this function is given by the power to *not saturate*.

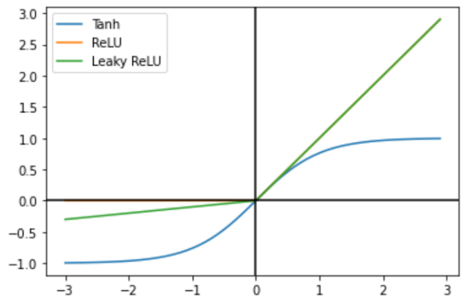


Fig. 5. Tanh, ReLU respectively Leaky ReLU on MNIST dataset

They have also shown that LReLU has a zero's sparsity for gradient which is more robust during the optimization. [8]

*D. ELU (**E**xponential **L**inear **U**nits)*

In 2015 Clevert *et al.* introduced a new activation function called **E**xponential **L**inear **U**nits (ELU) which reaches the training faster in deep neural networks and drives to high accuracy of classification. Also, ELU reduces the vanishing gradient, but its significant property is the power to have negative values which permit to push average activations of the units closer to 0, such as normalization but with a smaller computational complexity. [9]

The remarkable disadvantage of this function consists of exponential computation, how we can see in its equation:

$$f(x) = \begin{Bmatrix} x, \; if \; x > 0 \\ \alpha(\exp(x) - 1), \; if \; x \le 0 \end{Bmatrix} \tag{5}$$

In recent work, Yuandong Tian from Facebook AI Research showed that if the activations of layer 1 start to align, then activations of layer 2, which depends on activations of layer 1, this creates a critical path from important nodes at the lowest layer all the way to the output, and this critical path accelerates the convergence of that node. [10]

E. *Self-Normalizing Neural Networks (**S**elf-Normalizing **E**xponential **L**inear **U**nits)*

In [11] Günter Klambauer *et al.* introduced a new type of neural network called Self-Normalizing neural networks (SNN) for abstract representations at a high level. They also developed a new function called *Self-Normalizing Neural Networks*, which introduces self-normalizing properties. These properties permit:

- Training deep neural networks with more layers.
- Using a strong regularization.
- Making the learning very robust.

F. *Softmax*

Softmax function limits the output's function in the range [0,1]. It means that the output can be interpreted as a probability. In other words, Softmax functions are sigmoid functions with more classes, which means that they are used to determine the probability of more classes simultaneously.

Usually, the Softmax function is the final layer in neural networks. The most important remark consists of its property that softmax layers should have the same number as the output layer.

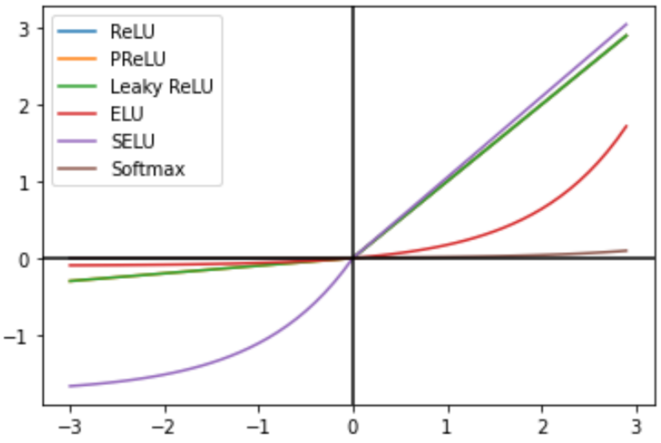
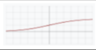
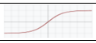
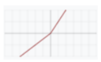
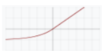
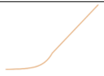
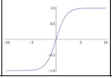


Fig. 6. A comparison of current functions on MNIST dataset

As you saw above in Figure 6, we plotted the current functions which we presented in this paper, more exactly: ReLU, PReLU, Leaky ReLU, ELU, SELU, and Softmax. In Table 1 below you can see the most remarkable advantages and disadvantages of classic and current activation functions. Each new activation function brings an improvement in solving issues that their predecessor functions have.

TABLE I. ADVANTAGES AND DISADVANTAGES OF ACTIVATION FUNCTIONS

Type	Activation function	Plot	Range	Monotonic	Monotonic and derivative	Approximates identity near the origin	Advantages	Disadvantages
Classic	Sigmoid		((0,1)	Yes	No	No	Not blowing up activation	Tend to vanish gradient Expensive computationally
	Tanh		(-1,1)	Yes	Yes	Yes	Its derivative gets more value Has a wider range for faster learning	Vanishing gradient
	ReLU		[0,+∞)	Yes	No	No	More computationally efficient Tend to show better convergence performance than sigmoid network will run faster	Tend to blow up activation Dying Relu problem Losing information on negative side because it
	PReLU		(-∞,+∞)	Yes, if α≥0	Yes, if α=0	Yes, if α=1	The negative region of learning to provide the values Has learnable parameter The using PReLu's came at a marginal computational cost	It saturates for large negative values, allowing them to be essentially inactive.
	Leaky ReLU		(-∞,+∞)	Yes	No	No	A bit balanced (negative values near 0)	Time training model is low Unlike PreLU, the coefficient of x is predefined and Neural network can not decide its value
	ELU		(-α,+∞)	Yes, if α≥0	Yes, if α=1	Yes, if α=1	Doesn't have the dying ReLU problem The function tends to converge cost to zero faster and produces more accurate results	Saturates for large negative values
	SELU		(-λα,+∞)	Yes	No	No	Is some kind of ELU α and λ are two fixed parameters, meaning we don't backpropagate through them Seems like SELU converges better and gets better accuracy on the test set on MNIST	SELU can't make it work alone, so a custom weight initialization technique is being used
Current	Softmax		(-1,1)	Yes	Yes	Yes	Very similar to Sigmoid function, the most important difference is that it is preferred in the output layer of deep learning models, especially when it is necessary to classify more than two very fast to train and predict.	It will not work if your data is not linearly separable It does not support null rejection

IV. Conclusions

The sigmoid function is the best choice in classification for its applicable manner and **M**ean **S**quared **E**rror (MSE). Also, in [12] it showed that the performance is impacted by dataset dimension, number of classes and time spent for training.

In [13] M. Mercioni *et al.*, showed that in the case of a neural architecture without hidden layers, using the sigmoid function as activation function, it observed that the value of β coefficient decreases if the error increases during an epoch.

The current error of neuron compares with the previous error of the same neuron. Depending on this comparison, β parameter changes asynchronous and it keeps the current error for next epochs for an increase of performance.

Also [14] C. T. Chen *et al.*, proved that an adaptive sigmoid function with a single hidden layer drives to better modeling of data. As the following overview of activation functions, some of them parametric or non-parametric we can conclude that the suitable activation functions depend on architecture network, how deep is the network, and for what task it is recommended.

As future work, we will continue with the impact study of classic versus current activation functions on newest and powerful datasets like Fashion-MNIST, Cifar-10, Cifar-100, and so on for different types of tasks.

References

[1] Goodfellow *et al.*, Deep Learning, MIT Press, 2016.

[2] S. Hayou *et al.*, On the Impact of the Activation Function on Deep Neural Networks Training, arXiv:1902.06853v2 [stat.ML] , 2019.

[3] S. Siegelmann *et al.*, Turing computability with neural networks, 1991.

[4] C. Özkan *et al.*, The Comparison of Activation Functions for Multispectral Landsat TM Image Classification, Istanbul Photogrammetric Engineering & Remote Sensing, Vol. 69, No. 11, pp. 1225–1234, 2003.

[5] X. Glorot *et al.*, Deep Sparse Rectifier Neural Networks, Montreal.

[6] K. He *et al.*, Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, arXiv:1502.01852 [cs.CV], 2015.

[7] O. Russakovsky *et al.*, Imagenet large scale visual recognition challenge, arXiv:1409.0575, 2014.

[8] A. Maas *et al.*, Rectifier Nonlinearities Improve Neural Network Acoustic Models, Atlanta, Georgia, USA: Proceedings of the 30 th International Conference on Machine Learning, JMLR, 2013.

[9] O. Abiodun *et al.*, State-of-the-art in artificial neural network applications: A surve, https://doi.org/10.1016/j.heliyon.2018.e00938, 2018.

[10] Y. Tian, Over-parametrization as a catalyst for better generalization of deep ReLU Network, Facebook AI Research arXiv:1909.13458v1 [cs.LG] , 30 Sep 2019.

[11] S. Klambauer *et al.*, Self-Normalizing Neural Networks, In Advances in Neural Information Processing Systems (NIPS), 2017.

[12] A. Shenouda *et al.*, A Quantitative Comparison of Different MLP Activation, J. Wang *et al.* (Eds.): ISNN 2006, LNCS 3971, pp. 849 – 857, 2006.

[13] M. Mercioni *et al.*, Dynamic modification of Activation Function using the Backpropagation Algorithm in the Artificial Neural Networks, (IJACSA) International Journal of Advanced Computer Science and Applications, Volume 10 No. 4, 2019.

[14] C. Chen *et al.*, A Feedforward Neural Network with Function Shape Autotuning, Neural Networks, Vol. 9, pp. 627-641, 1996.