CSCanada
Management Science and Engineering
Vol. 14, No. 1, 2020, pp. 53-58
DOI:10.3968/11667

ISSN 1913-0341 [Print]
ISSN 1913-035X [Online]
www.cscanada.net
www.cscanada.org

Improvements on Activation Functions in ANN: An Overview

YANG Lexuan[a],*

[a]Hangzhou No.2 High School of Zhejiang Province; Hangzhou, China.
*Corresponding author.

Received 9 April 2020; accepted 21 May 2020
Published online 26 June 2020

Abstract

Activation functions are an essential part of artificial neural networks. Over the years, researches have been done to seek for new functions that perform better. There are several mainstream activation functions, such as sigmoid and ReLU, which are widely used across the decades. At the meantime, many modified versions of these functions are also proposed by researchers in order to further improve the performance. In this paper, limitations of the mainstream activation functions, as well as the main characteristics and relative performances of their modifications, are discussed.

key words: Activation functions; ANN; Piecewise-linear

Yang, L. X. (2020). Improvements on Activation Functions in ANN: An Overview. *Management Science and Engineering*, *14*(1), 53-58. Available from: URL: http://www.cscanada.net/index.php/mse/article/view/11667
DOI: http://dx.doi.org/10.3968/11667

INTRODUCTION

In 1943, Warren S. McCullock and Walter Pitts proposed the possibility of simulating neural activities with computational network models, and proved that artificial neural networks can theoretically approximate any function, either arithmetic or logical (Mcculloch & Pitts, 1943). The Artificial Neural Network (ANN) algorithm has since been further developed and applied in various fields, which has actively promoted the development of artificial intelligence.

An ANN is comprised of computable units called neurons (or nodes). The activation function in a neuron processes the inputted signals and determines the output. Activation functions are particularly important in the ANN algorithm; change in activation functions can have a significant impact on network performance. Therefore, in recent years, researches have been done in depth on the improvement of activation functions to solve or alleviate some problems encountered in practices with "classic" activation functions. This paper will give some brief examples and discussions on the modifications and improvements of mainstream activation functions.

1. INTRODUCTION OF ACTIVATION FUNCTIONS

Activation functions are functions that determine the output of the nodes (neurons) in ANNs. C. Gulcehre et al. pointed out that an activation function should be differentiable almost everywhere (Gulcehre, C., et al, 2016). Also, in order for ANNs to be capable of learning, an activation function must be nonlinear on its domain, or piecewise-linear.

An ANN with such activation function(s) is theoretically sufficient to approximate any function. Kur Hornik et al. pointed out that "there is a single hidden layer feedforward network that approximates any measurable function to any desired degree of accuracy on some compact set K." (Hornik, Stinchcombe, & White, 1989)

2. IMPROVEMENTS BASED ON SIGMOID FUNCTION

Sigmoid function is defined by

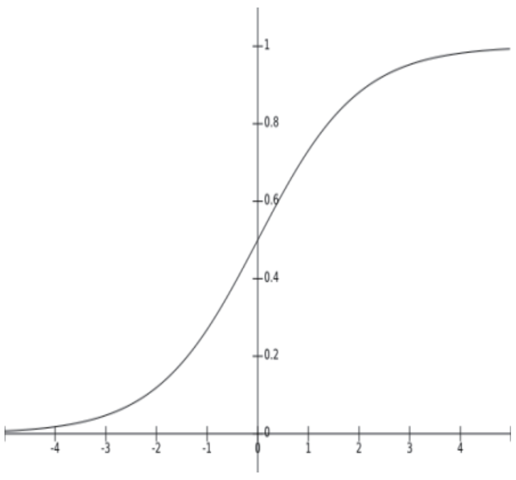

Figure 1
The graph of sigmoid function

The graph of the function is an S-shaped symmetric curve centred at (0,0.5), with an output range limited in (0,1). The derivative of the function is

The limitations of sigmoid are as follows:

Sigmoid reaches soft saturation at both sides. When the input falls into the saturated region, the activation values will be largely zeros, and therefore the network may not be trained properly.

The backpropagation algorithm computes gradients by the chain rule. In each iteration of training, the gradient is multiplied by the derivative of the activation function, . The derivative of sigmoid is always smaller than 1; therefore, the gradient will decrease exponentially in backpropagating, approaching towards zero. This is often called the *vanishing gradient problem*. Such problem will result in very small gradients in the front layer, meaning that when a gradient-descent based iterative method (e.g. stochastic gradient descent) is applied, the weight update in the front layer will be extremely slow, and the training will therefore be inefficient.

It is worth noting that *Tanh function*, another widely used S-shaped function, is often compared with sigmoid.

Tanh function is defined by

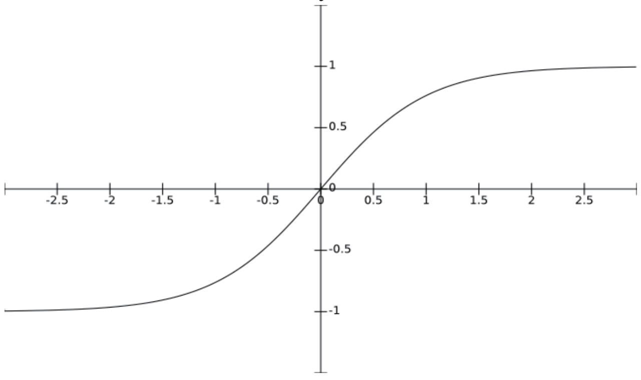

Figure 2
The graph of tanh function

It is very similar in nature to sigmoid, and therefore has similar drawbacks as the former. However, it is often proved to perform better than sigmoid. A possible reason is that, with an output range of (-1,1), it has a zero mean value, and therefore prevents *bias shift*, resulting in faster learning rate (Goodfellow, Bengio, & Courville, 2016; Glorot, Bordes, & Bengio, 2011; Clevert, Unterthiner, & Hochreiter, 2015).

An example of modifications based on sigmoid is the *bi-parametric sigmoid function* proposed by Huang et al. (Huang, Y., et al, 2017). Two independent parameters α and λ are introduced in the function and its derivative:

The parameter α () is introduced to prevent excessive input values from falling into the right-side saturated region, with λ () to countermand the vanishing effect of α on the gradient. Based on MNIST dataset, Huang et al. performed number classification tasks using deep belief network, benchmarking bi-parametric sigmoid against the ordinary sigmoid function. It turns out that compared with sigmoid, bi-parametric sigmoid can efficiently alleviate the vanishing gradient problem, therefore achieving better training results and faster convergence speed.

3. IMPROVEMENTS BASED ON RELU FUNCTION

Rectified Linear Unit (also *rectifier*, or ReLU) *function* is defined by

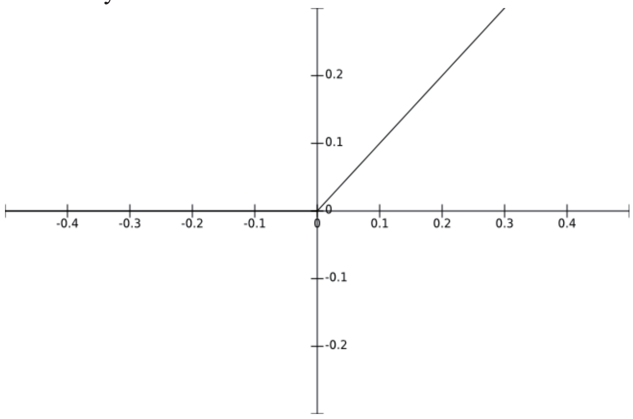

Figure 3
The graph of ReLU function

As X. Glorot et al. pointed out, in human brains, information stores in neurons *sparsely*, i.e. the percentage of neurons active at the same time is small (Glorot, Bordes, & Bengio, 2011). The fact that ReLU returns zero for all negative input allows the function to simulate this nature of biological neurons. Also, the gradient of ReLU when is always zero, so can efficiently prevent the vanishing gradient, compared with sigmoid and tanh, which have a much wider saturated region. It has been seen in practice that ReLU generally outperforms both sigmoid and tanh (Maas, et al, 2013).

The limitations of ReLU are as follows:

It *hard-saturates*, i.e. its gradient disappears, when, returning zero for all input values. As a result, if the input falls into this region, causing *neuron death*: weights of the neurons can no longer be updated. The network may therefore be completely incapable of learning. (it is worth noting that, when comparing ReLU with softplus function, Glorot et al. discovered that *supervised* training can in fact be benefited by hard-saturation of ReLU, providing the gradient *can* backpropagate, i.e. neurons in the same layer are not deactivated in the same time (Glorot, Bordes, & Bengio, 2011)

ReLU has a positive mean value, and non-zero mean will cause *bias shift*: the mean activation value will propagate to next layer as bias, deviating the standard gradient from the natural gradient. Oscillation will occur due to bias shift, and will therefore cause low learning rate.

ReLU is unrestrained in the positive axis. As P. Ramachandran et al. pointed out, such nature of ReLU means that it will not saturate when , which is desirable (Ramachandran, Zoph, & Le, 2017); however, this also means the gradient could easily explode if excessive activation values occur.

In order to solve these problems, many modifications based on ReLU are proposed. A. L. Maas et al. introduced *Leaky ReLU function* (Maas, et al, 2013) in 2013, which is defined by

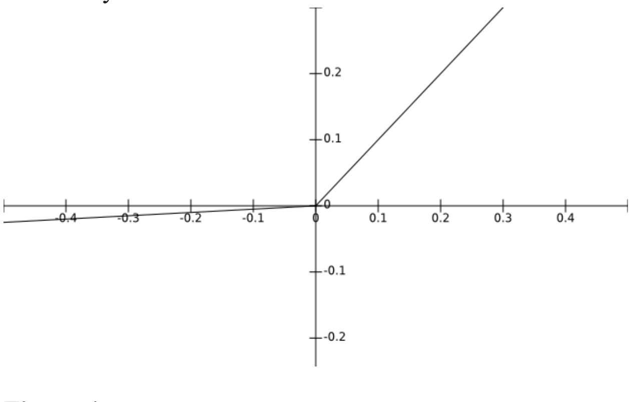

Figure 4
The graph of Leaky ReLU

The hyperparameter is generally set at 0.01. allows the function to have a slight gradient on (see Fig.4), so as to avoid neuron death. Based on 300-hour Switchboard conversational telephone speech corpus (LDC97S62), Maas et al. performed LVCSR experiments. They discovered that there was no significant improvement on network performance when replacing ReLU with Leaky ReLU, and concluded that the said problem of ReLU will not adversely affect training. Followed researches shows that Leaky ReLU can in practice improve network performance, and, generally, the improvement is more significant with a greater value of (Xu, B., et al, 2015).

K. He et al. proposed *Parametric ReLU* (PReLU) *function* in 2015 (He, et al, 2015), whose basic form is the same as Leaky ReLU, but the parameter is obtained through learning. Based on ImageNet 2012 dataset, He et al. performed image recognition experiments, reaching the conclusion that PReLU generally outperforms ReLU without increasing computing cost.

In order to solve the bias shift problem of ReLU, D.-A. Clevert et al. proposed *Exponential Linear Unit* (ELU) *function* in 2016 (Clevert, Unterthiner, & Hochreiter, 2015), which is defined by

Its derivative is

where is a hyperparameter. The mean value of ELU is closer to zero than that of ReLU, therefore alleviating bias shift to some extent; at the meantime, according to Clevert et al., because functions like Leaky ReLU which also have negative values are not saturated in the negative region, neurons may be activated by noise; ELU, on the other hand, has a soft-saturated region, making it more robust to noise (see Fig. 5) (He, et al., 2015). Also, by introducing output values when , neuron death can also be prevented. Based on MNIST, CIFAR-10, CIFAR-100 and ImageNet datasets, the team benchmarked the performance of ELU against other ReLUs such as Leaky ReLU and PReLU, and observed that ELU generally outperformed the others.

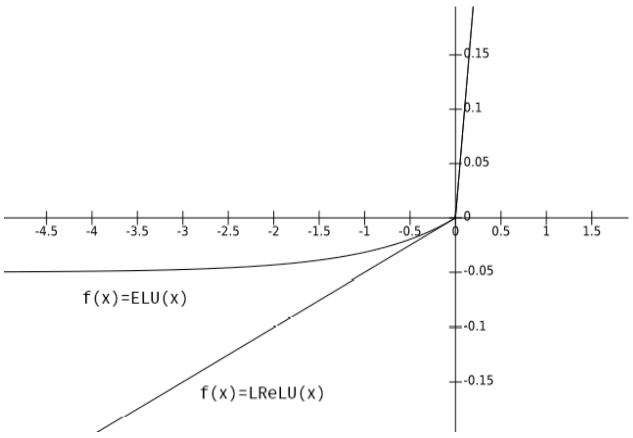

Figure 5
Comparing the graphs of Leaky ReLU and ELU, . Note that ELU has "a clear saturation plateau in its negative region", as the team put it (He, et al, 2015)

There are many other improving attempts based on ReLU. Several examples are as follows:

In order to prevent neuron death, and the problem that *ReLU-Softplus* (a hybrid function defined by , proposed by Q. Shi (Shi, 2017) only converges under the learning rate of no greater than 0.0001, H. Wang et al. proposed *ReLU-Softsign function* (WANG, et al, 2019), which is defined by

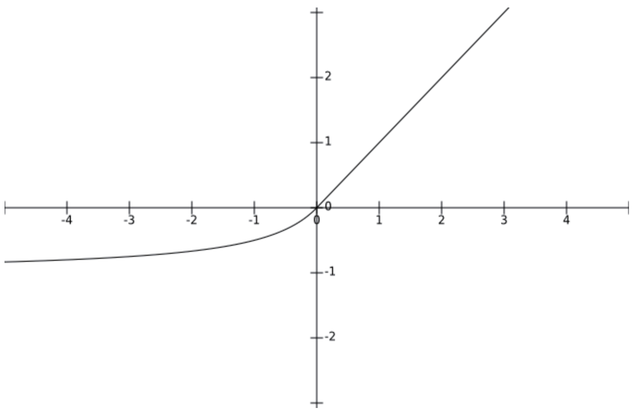

Figure 6
The graph of ReLU-Softplus

Based on MNIST, PI100, CIFAR-100 and Caltech256 datasets, Wang et al. performed image classification experiments. It is discovered that ReLU-Softplus achieves higher precision and faster convergence speed than several existing activation functions.

In order to prevent vanishing gradient and neuron death, and to improve noise robustness of the network, T. Zhang et al. proposed *TReLU* (Zhang, Yang, Song, & Song, 2019) *function*, a hybrid of tanh and PReLU, defined by

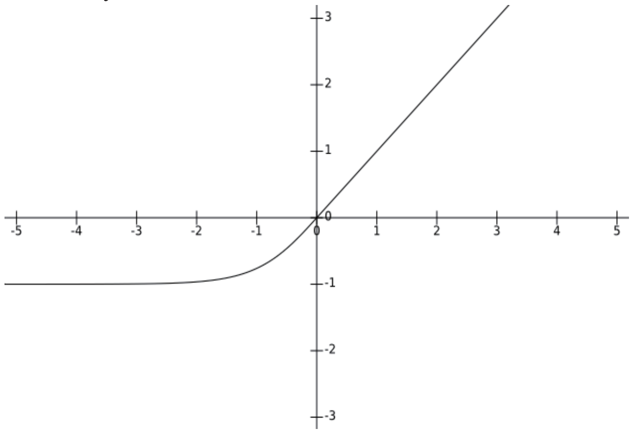

Figure 7
The graph of TReLU

TReLU is very similar to ELU, whose negative region is soft-saturated to make it noise robust. Based on MNIST, CIFAR-10 and NWPU-RESISC45 datasets, Zhang et al. performed experiments and observed that TReLU outperformed both ReLU and PReLU.

In order to prevent explosion, X. Liu et al. proposed *threshold ReLU function* (Liu, Guo, & Li, 2019). is set to zero after reaches a threshold value , so that the network stops learning in the region. Based on Caltech 101 and Caltech 256 datasets, Liu et al. performed image classification experiments on Alexnet and threshold ReLU, benchmarking against ReLU. They concluded that threshold ReLU outperformed ReLU in precision and extent of convergence. Optimal performance occurred when .

To prevent bias shift, vanishing gradient and explosion, Y. He et al. proposed *Tanh ReLU function* (He, Cheng, Zhang, & Li, 2019), which is defined by

It is easy to see that TReLU and Tanh ReLU are very similar, but a threshold value is introduced in Tanh ReLU to prevent explosion. He et al. performed classification experiments on MNIST dataset, benchmarking Tanh ReLU against ReLU. Tanh ReLU generally outperformed ReLU according to the team. Particularly, when the learning rate is reduced to 0.001, the network performance is greatly improved.

In order to prevent vanishing gradient and neuron death, B. Xu et al. proposed *ArcReLU function* (Xu & Xu, 2019), a hybrid of Arctan and ReLU, which is defined by

It is clear that its derivative is always positive, so a single-layer net will be a convex function. The team predicted that the network would therefore have better convergence. Xu et al. performed five classification experiments to benchmark ArcReLU against ReLU and ELU; the dataset that each experiment used was respectively Pima Indians Diabetes Database, The Iris Dataset, Car Evaluation Dataset, US Census Income Dataset and Avila Dataset. According to Xu et al., ArcReLU "[…] can not only significantly accelerate the training speed of BP neural network, but also effectively reduce the training error and avoid the problem of gradient disappearance." (Xu & Xu, 2019)

4. IMPROVEMENTS BASED ON SOFTPLUS FUNCTION

Softplus function is defined by

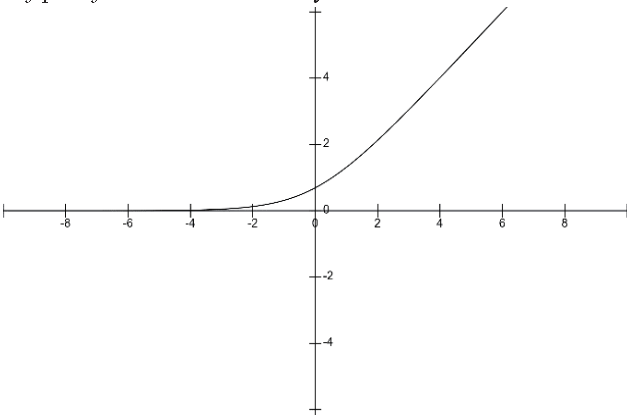

Figure 8
The graph of softplus function

Its graph is very similar to ReLU, but smooth around the origin. Therefore, softplus and ReLU has similar advantages and drawbacks. Compared with ReLU, softplus does not hard-saturate when , so it can theoretically prevent the gradient from diminishing completely for all negative .

An example of modifications based on softplus is *Rand Softplus function* (Mai, Chen, & Zhang, 2019) proposed by Y. Mai et al., defined by

where parameter is introduced to reflect the randomness of biological neurons, so as to improve noise robustness of softplus networks. The value of is determined by

Calculating the standard deviations of noise;

Calculating mean standard deviation of each layer, and reset 's to when ;

Normalizing the deviations to (0,1) to obtain .

A threshold value is introduced to achieve sparsity: values of smaller than is reset to zero. Based on CK+ (The Extended Cohn-Kande Dataset), KDEF (Karolinska Directed Emotional Faces) and GENKI-4K datasets, Mai et al. performed facial expression recognition experiences using deep residual networks. They concluded that compared with mainstream activation functions, Rand Softplus performed better for noisy input.

5. IMPROVEMENTS BASED ON SWISH FUNCTION

Swish function was proposed by Ramachandran et al. in 2017 (Ramachandran, Zoph, & Le, 2017), which is defined by

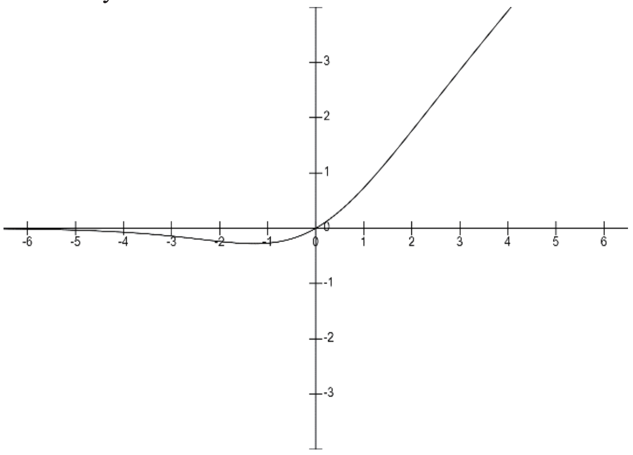

Figure 9
The graph of Swish

The graph crosses the origin, and as . It can also be seen as a "smooth ReLU", but unlike softplus, it is non-monotonic, which is likely to be "advantageous" according to Ramachandran et al. (Ramachandran, Zoph, & Le, 2017) The negative activation value can also reduce neuron death. Based on MNIST, CIFAR and ImageNet datasets, the team benchmarked Swish against softplus and ReLU functions such as Leaky ReLU, PReLU and ELU, and concluded that Swish "robustly" outperformed the others.

An example of modifications based on Swish is *Xwish function* proposed by Y. Liu et al. (Liu, Wang, & Xu, 2019), which is defined by

The parameter controls the speed for the derivative to converge to 0 or 1. It is clear that the group draws inspiration from the basic form of Swish, . Based on MINST and Cifar-10 datasets, Liu et al. performed experiments under Tensorflow framework, benchmarking Xwish against ReLU, LReLU and tanh. They observed that Xwish network showed the best convergence compared with the other three.

CONCLUSION

In view of the shortcomings of mainstream activation functions, many researchers have proposed modified versions of existing functions. A considerable proportion of researches are devoted to solving the problems commonly existing in mainstream activation functions (e.g. traditional S-shaped functions and ReLUs) such as vanishing gradient and neuron death. There are also studies focusing on improving noise robustness, extent of convergence, and stability.

It is also worth noting that ReLU gained growing popularity due to its promising performance. Most improved functions in this article (including the ReLU-based piecewise functions, softplus and Swish) are based on ReLU. Some studies suggest that behind the success of ReLU are its *unboundedness* when and its *sparse activation*. Finding the actual reason *why* these characteristics are advantageous may help to select the activation function applicable in different application scenarios and to propose better activation functions, thereby promoting the further development of ANN.

REFERENCES

Clevert, D.-A., Unterthiner, T., & Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). *ArXiv preprint arXiv:1511.07289*.

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. *Proceedings of the fourteenth international conference on artificial intelligence and statistics*.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning* (p.195). Cambridge: The MIT Press.

Gulcehre, C., et al. (2016). *Noisy Activation Functions*.

He, K. M., et al. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *Proceedings of the IEEE international conference on computer vision*.

He, Y., Cheng, L. F., Zhang, P. L., & Li, Y. (2019). Application of activation function of deep neural network. *Measurement and control technology, 38*(4), 50-53.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks, 2*(5), 359-366.

Huang, Y., et al. (2017). A study of traning algorithm in deep neural networks based on sigmoid activation function. *Computer Measurement & Control, 2,* 132-135.

Liu, X. W., Guo, D. B., & Li, C. (2019). An Improvement of the Activation Function in Convolutional Neural Networks. *Journal of Test and Measurement Technology, 33*(2), 121-125.

Liu, Y. Q., Wang, T. H., & Xu, X. (2019). New adaptive activation function of deep learning neural networks. *Journal of Jilin University (Science Edition), 57*(4), 857-859.

Maas, A. L., Hannun, A. Y., & Andrew Y. Ng. (2013). Rectifier nonlinearities improve neural network acoustic models. *Proc. icml*., *30*(1), 2013.

Mai, Y. C., Chen, Y. H., & Zhang, L. (2019). Bio-inspired activation function with strong anti-noise ability. *Computer Science*, (7), 206-210.

Mcculloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology, 52*(1-2), 99-115.

Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Swish: a self-gated activation function. *ArXiv preprint arXiv:1710.05941*.

Shi, Q. (2017). *Research and verification of image classification optimization algorithm based on convolutional neural network.* Beijing: Beijing Jiaotong University.

WANG, H. X., Zhou, J. Q., GU, C. H., & Lin, H. (2019). Design of activation function in CNN for image classification. *Journal of Zhejiang University (Engineering Science), 53*(7), 1363-1373 DOI: 10.3785/j.issn.1008-973X.2019.07.016

Xu, B., et al. (2015). Empirical evaluation of rectified activations in convolutional network. *ArXiv preprint arXiv:1505.00853*.

Xu, Y. J., & Xu, F. F. (2019). Optimization of activation function in neural network based on ArcReLU function. *Journal of Data Acquisition & Processing,* (3), 15.

Zhang, T., Yang, J., Song, W. A., & Song, C. F. (2019). Research on improved activation function TReLU. *Journal of Chinese Computer Systems, 40*(1), 58-63.