# Activation Functions and Their Characteristics in Deep Neural Networks

Bin Ding, Huimin Qian, Jun Zhou
College of Energy and Electrical Engineering, Hohai University, Nanjing, 211100
E-mail: 774738312@qq.com, qhmin0316@163.com, zhouj@lzu.edu.cn

**Abstract:** Deep neural networks have gained remarkable achievements in many research areas, especially in computer vision, and natural language processing. The great successes of deep neural networks depend on several aspects in which the development of activation function is one of the most important elements. Being aware of this, a number of researches have concentrated on the performance improvements after the revision of a certain activation function in some specified neural networks. We have noticed that there are few papers to review thoroughly the activation functions employed by the neural networks. Therefore, considering the impact of improving the performance of neural networks with deep architectures, the status and the developments of commonly used activation functions will be investigated in this paper. More specifically, the definitions, the impacts on the neural networks, and the advantages and disadvantages of quite a few activation functions will be discussed in this paper. Furthermore, experimental results on the dataset MNIST are employed to compare the performance of different activation functions.

**Key Words:** neural network, deep architecture, activation function

## 1 INTRODUCTION

In the past few years, tremendous improvements have been witnessed in the representation and recognition performance of neural networks with deep architectures [1, 2], due to which, the landscapes of computer vision and natural language processing have been noticeably changed [3–5]. The revolutionary changes are resulted from the next crucial elements including building more powerful models [4, 19–22, 25], accumulating larger-scale dataset [19, 21], developing higher-performance hardware, designing more effective regularization techniques [21, 21, 23], and so on. Among which, the developments of activation functions also have played a vital role of improving the performance of deep neural networks, thus, more and more efforts have concentrated on the study of activation functions [6–12]. Since Nair and Hinton [7] first proposed the rectified linear units to improve the performance of Restricted Boltzmann Machines, saturated activation function, such as sigmoid and tanh, are replaced by non-saturated counterpart, such as ReLU, ELU, to solve the so-called vanishing gradient and to accelerate the convergence speed in the neural networks with deep architectures, like convolutional neural network, recurrent neural network.

In order to understand the status and performance improvement of activation function in deep neural networks thoroughly, the definition, pros and cons of commonly used activation functions will be discussed in this paper. And the comparison of experimental results on MNIST dataset will be illustrated as well.

This work is supported by the Natural Science Foundation of Jiangsu Province under Grant BK20140860 and the Natural Science Foundation of China under Grant 61573001.

## 2 ACTIVATION FUNCTION

Artificial neural network is the analog of biological neural network. Therefore, the primary task of constructing an artificial neural network is to design the artificial neuron model since neurons are the basic units of biological neural network. Next, briefly explain the operating principle of a biological neuron. A biological neuron receives the electrical signals sent by other neurons with different weights. When the synthetic value of all the electrical signals is big enough to stimulate the neuron, it turns into the excited state to output a response. Otherwise, it keeps the inactive state.

According to the principle, the artificial neuron model is designed. Fig. 1 is the schematic diagram of an artificial neuron (AU), where $\{x_1, x_2, \cdots, x_n\}$ are the inputs of an AU, $\{w_1, w_2, \cdots, w_n\}$ are the weights corresponding to the inputs, $b$ is the bias, and the addition unit $\sum$ gets the linear weight sum $z$ of the inputs $\{x_1, x_2, \cdots, x_n\}$ and the bias. Denote $x = [x_1, x_2, \cdots, x_n] \in R^n$ and

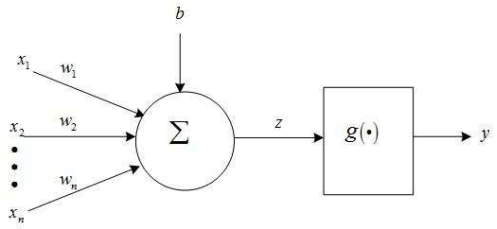


Figure 1: The schematic diagram of an artificial neuron

$w = [w_1, w_2, \cdots, w_n] \in R^n$, then the output $z$ of the addition unit can be represented as

$$z = xw^T + b$$

The function $g(\cdot)$, which is named as activation function, is used to simulate the response state of biological neuron and obtain the output $y$, that is

$$y = g(z)$$

Caglar et. al. gave the definition of an activation function in [13], which is " An activation function is a function $g : R \longrightarrow R$ that is differentiable almost everywhere."
Nonlinear activation functions have brought neural networks their nonlinear capabilities, which means neural networks can differentiate those data that can not be classified linearly in the current data space.
The definition given by Caglar [13] shows that the continuous differentiable function can be used as an activation function. Whereas, it is not always the truth to the deep neural networks due to the difficulties in training. Next, the commonly used activation functions will be analyzed thoroughly.

### 2.1 Sigmoid and Its Improvements

#### 2.1.1 Sigmoid

Sigmoid function is one of the most common forms of activation functions. The definition of sigmoid function is as follows,

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

in which $x \in (-\infty, +\infty)$, $g \in (0, 1)$, as seen in Fig. 2. It

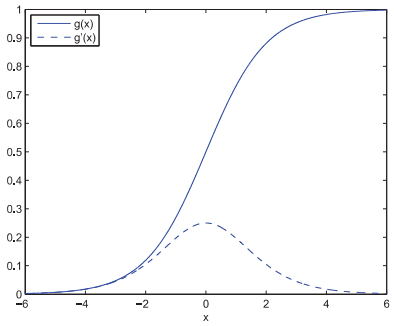


Figure 2: The graphic depiction of Sigmoid function and its derivative

is known that the modeling and training process of a multilayer neural network can be divided into two parts: forward propagation and back propagation. And in the back propagation, the derivatives of activation functions in each layer should be calculated. The sigmoid function is a continuous function, which means that it is differentiable everywhere. The derivatives of sigmoid function

$$g'(x) = \frac{e^{-x}}{(1 + e^{-x})^2}$$

is easy to be calculated. Therefore, the sigmoid function was commonly used in shallow neural networks. In addition, the sigmoid function is frequently employed in the output level of a neural network owing to its value distribution. While sigmoid function is rarely adopted in the deep neural networks except the output level since its saturation. Exactly, it is soft saturate since it only achieves zero gradient in the limit [13]

$$\lim_{x \to +\infty} g'(x) = 0$$

and

$$\lim_{x \to -\infty} g'(x) = 0$$

as seen in Fig. 2. The soft saturation results in the difficulties of training a deep neural network. More specific, in the process of optimizing loss function, the derivatives of sigmoid function, which is contributed to update the weights and the bias, will reduce to zero when it comes to the saturation area, which brings about the less contributions of the first several layers in the knowledge learning from training samples. In fact, the situation is called vanishing gradient. Generally, the vanishing gradient will emerge in a neural network with more than five layers [6].
According to the limitation of Sigmoid function, several improvements have been proposed. Huang et al. [14] introduced double-parameters to sigmoid function. One of the parameter is used to generate an appropriate input $z$ which can not lead the input to the saturation area; the other is to control the decay of residual error. Caglar et al. [13] proposed to inject the noise to the activation functions which makes the loss function is easier to optimize. Another development is hyperbolic tangent function, which will be illustrated next.

#### 2.1.2 Hyperbolic Tangent

Hyperbolic tangent function can be easily defined as the ratio between the sine and the cosine functions.

$$tanh(x) = \frac{sinh(x)}{cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2}$$

It is similar to sigmoid function and can be deducted from the sigmoid function in (1) as follows

$$tanh(x) = 2sigmoid(2x) - 1$$

where $sigmoid(\cdot)$ is $g(x)$ in equation (1). The hyperbolic tangent function ranges outputs between -1 and 1 as seen in Fig. 3. It is also a continuous and monotonic function, and it is differentiable everywhere. It is symmetric about the origin (see Fig. 3), of which the outputs, namely the inputs of next layer, are more likely on average close to zero. Therefore, the hyperbolic tangent functions are more preferred than sigmoid functions. In addition, neural networks with hyperbolic tangent activation functions converge faster than those with sigmoid activation functions [15]. And the neural networks with hyperbolic tangent activation functions have lower classification error than those with sigmoid activation functions [6].
However, the calculation of the derivatives of hyperbolic tangent functions, listed as follows,

$$tanh'(x) = 2sigmoid'(2x) - 1 = \frac{4e^{-2x}}{(1 + e^{-2x})^2}$$

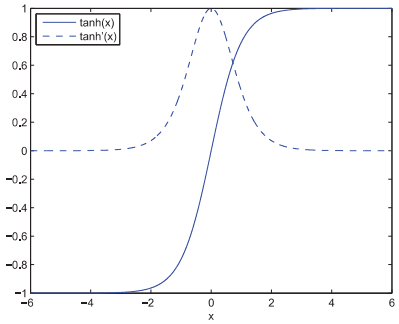


Figure 3: The graphic depiction of hyperbolic tangent function and its derivatives.

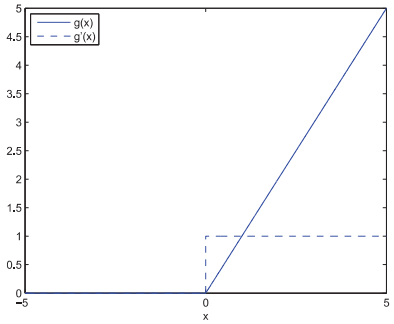


Figure 4: The graphic depiction of ReLU function and its derivatives.

is more complicated than sigmoid function. And it has the same soft saturation as sigmoid function, which also has the vanishing gradient problem.

### 2.2 ReLU and Its Improvements

The neuroscience research found that cortical neurons are rarely in their maximum saturation regime, and suggests that their activation function can be approximated by a rectifier [29]. That is to say, the operating mode of the neurons has the characteristic of sparsity. More specific, only one percent to four percent of neurons in the brain can be activated simultaneously. However, in the neural networks with sigmoid or hyperbolic tangent activation functions, almost one half of the neuron units are activated at the same time, which is inconsistent with the research in neuroscience. Furthermore, activating more neuron units will bring about more difficulties in the training of a deep neural network. Rectified linear units (ReLU), firstly introduced by Nair and Hinton for Restricted Boltzmann machines [7], will help hidden layer of the neural network to obtain the sparse output matrix, which can improve the efficiency.
ReLU function and its improvements are almost the most popular activation functions used in deep neural networks currently [6, 16, 24, 26]. It is said that although the main difficulty of training deep networks are resolved by the idea of initializing each layer by unsupervised learning, while the employment of ReLU activation functions can also be seen as a breakthrough in directly supervised training of deep networks.

#### 2.2.1 ReLU

The definition of ReLU is as follows

$$g(x) = max(0, x) = \begin{cases} x & if \ \ x \geq 0 \\ 0 & if \ \ x < 0 \end{cases} \tag{3}$$

The graph is depicted in Fig. 4. ReLU activation function is non-saturated, and its derivative function is

$$g'(x) = \begin{cases} 1 & if \ \ x \geq 0 \\ 0 & if \ \ x < 0 \end{cases}$$

which is a constant when the input $x > 0$. Thus, the vanishing gradient problem can be released. Specifically, ReLU function has the following advantages [6, 16]:

- Computations of neural networks with ReLU functions are cheaper than sigmoid and hyperbolic tangent activation functions because there are no need for computing the exponential functions in activations.
- The neural networks with ReLU activation functions converge much faster than those with saturating activation functions in terms of training time with gradient descent.
- The ReLU function allows a network to easily obtain sparse representation. More specific, the output is 0 when the input $x < 0$, which provides the sparsity in the activation of neuron units and improves the efficiency of data learning. When the input $x \geq 0$, the features of the data can be retained largely.
- The derivatives of ReLU function keep as the constant 1, which can avoid trapping into the local optimization and resolve the vanishing gradient effect occurred in sigmoid and hyperbolic tangent activation functions.
- Deep neural networks with ReLU activation functions can reach their best performance without requiring any unsupervised pre-training on purely supervised tasks with large labeled datasets.

However, the derivatives $g'(x) = 0$ when $x < 0$ so the ReLU function is left-hard-saturating. And the relative weights might not be updated any more and that leads to the death of some neuron units, which means that these neuron units will never be activated. Another disadvantage of ReLU is that the average of the units' outputs is identically positive, which will cause a bias shift for units in the next layer. The above two attributes both have a negative impact on convergence of the corresponding deep neural networks [17].

#### 2.2.2 LReLU, PReLU, and RReLU

The possible death of some neuron units of neural networks with ReLU functions are resulted from the compulsive operation of letting $g(x) = 0$ when $x < 0$. In order to alleviate this potential problem, Maas et al. [17] proposed the

leaky rectified linear units (LReLU), see in equation(4).

$$g(x) = max(0, x) = \begin{cases} x & if \ \ x \geq 0 \\ 0.01x & if \ \ x < 0 \end{cases} \tag{4}$$

Its derivative function is

$$g'(x) = \begin{cases} 1 & if \ \ x > 0 \\ 0.01 & otherwise \end{cases}$$

Fig. 5 gives the graph of LReLU function, which is nearly identical to the ReLU function in Fig. 4. As we can see

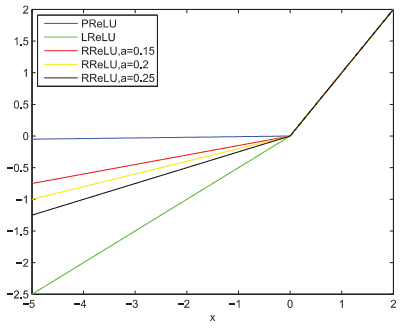


Figure 5: The graphic depiction of LReLU, PReLU, and RReLU function

that the LReLU allows for a small, non-zero gradient when the unit is saturated and not active. Therefore, there are no zero gradients and no neuron units can be "off" always.

It should be acknowledged that the sparsity has been lost when replacing ReLU with LReLU. Fortunately, the experimental results in [17] illustrated that the impact on the classification accuracy under the modification can not be recognized while the learning capabilities of the neural networks become more robust.

Furthermore, He et al. [12] presented the parametric rectified linear unit (PReLU) by replacing the constant $0.01$ of equation (4) with a learnable parameter. The definition of PReLU is

$$g(x) = \begin{cases} x & if \ \ x \geq 0 \\ ax & if \ \ x < 0 \end{cases} \tag{5}$$

where $a$ is a learnable parameter. The experiments in [12] show that LReLU can lead to better results than ReLU and PReLU by choosing the value $a$ very carefully, but it needs tedious, repeated training. On the contrary, PReLU can learn the parameter from the data. It is verified that the PReLU converges faster and has lower train error than ReLU. In addition, it is said that the introduce of parameter $a$ for the activation functions will not bring about over-fitting [12]. Wei et al. [18] has used deep convolutional neural network combining L1 regularization and PReLU activation function to the research on image retrieval, in which the experiments demonstrate that the over-fitting problem has indeed resolved and the efficiency of image retrieval has been improved by adopting PReLU activation functions.

Another improvement of ReLU, namely randomized rectified linear unit (RReLU), should be discussed. As we know, the slopes of the negative parts are set as a constant in LReLU activation functions, and as a learnable parameter in PReLU activation functions, respectively. While in RReLU [10] , the slopes are randomized in a given range in the training, and then fixed in the testing. The definition of RReLU is that

$$g(x) = \begin{cases} x & if \ \ x \geq 0 \\ ax & if \ \ x < 0 \end{cases} \tag{6}$$

where

$$a \sim U(A, B), A < BandA, B \in [0, 1)$$

In the training process, $a$ is a random number sampled from a uniform distribution $U(A, B)$. In the test process, the average of all the parameters $a$ in training are taken, and the parameter is set as $\frac{A+B}{2}$ in [10]. The performance of RReLU is investigated to be better than ReLU, LReLU and PReLU in specific experiments [10].

### 2.3 ELU and Its Improvements

In order to push the activation means of activation functions closer to zero to decrease the bias shift effect of ReLU, the exponential linear unit (ELU) with $\alpha > 0$ was proposed as follows [11].

$$g(x) = \begin{cases} x & if \ \ x > 0 \\ \alpha(e^x - 1) & if \ \ x \leq 0 \end{cases} \tag{7}$$

And its derivative function is

$$g'(x) = \begin{cases} 1 & if \ \ x > 0 \\ \alpha e^x & if \ \ x \leq 0 \end{cases}$$

The graphic depiction is shown in Fig. 6. The parameter $\alpha$

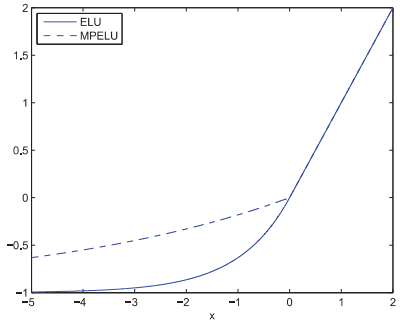


Figure 6: The graphic depiction of ELU and MPELU function

manages the value to which an ELU saturates for negative network inputs. The vanishing gradient problem is alleviated because the positive part of ELU is identity. More specific, the derivative is one when $x > 0$. The left-saturation lets deep neural networks with ELU activation functions become more robust to the input perturbation or noise.

The output average of ELU is approach to zero which contribute to faster convergence. Experimental results in [11] have shown that ELU can enable faster learning and the generalization performance of ELU are better than ReLU and LReLU activation functions when the layers of deep neural networks are more than five.
However, the same drawback of ELU as LReLU is that searching a reasonable $\alpha$ is important but time-consuming. According to this, Li et al. [8] proposed the multiple parametric exponential linear unit (MPELU).

$$g(x) = \begin{cases} x & if \ \ x > 0 \\ \alpha(e^{\beta x} - 1) & if \ \ x \leq 0 \end{cases} \tag{8}$$

Among which, $\alpha, \beta$ are learnable parameters which control the value to and at which MPELU saturates respectively. And it has been reported that MPELU can become ELU, ReLU, LReLU, or PReLU by adjusting the two parameters $\alpha, \beta$. Therefore, the advantages of MPELU include: 1) The convergence property of ELU is also possessed by MPELU, 2) The generalization capability of MPELU is better than ReLU and PReLU based on the experiments on ImageNet database.

## 3 EXPERIMENTS

In this paper, we have conducted experiments by deep convolutional neural network (DCNN), whose structure and parameters can be seen in Fig. 7. From the figure we can see that, the DCNN contains two 5*5 convolutional layers and two 2*2 max-pooling layers, both the stride is fixed to 1 pixel. Then followed a Fully-Connected(FC) layer. During training, the input to our DCNN is a fixed-size 28*28 gray image. The dataset is segmented as training dataset with $60,000$ samples and test dataset with $10,000$ samples. The cross-entropy loss function is used, the learning rate is chosen as $e^{-4}$, and the number of iterations is $20,000$.
The activation functions including sigmoid, hyperbolic tangent, ReLU, RReLU, and ELU are considered for the DCNN in Fig. 7 respectively. For RReLU and ELU, we conducted different experiments by adjusting the values of parameter $a$ for RReLU and $\alpha$ for ELU to choose their best performance. The classification errors with different activation functions are listed in Tab. 1. It can be seen that

Table 1: Classification error comparisons of DCNNs with different activation functions on MNIST

| Activation function | Parameter | Error (%) |
|---|---|---|
| Sigmoid | - | 1.15 |
| Tanh | - | 1.12 |
| ReLU | - | 0.8 |
| RReLU | $a = 0.5$ | 0.99 |
| ELU | $\alpha = 1$ | 1.1 |

the proposed network in this paper with ReLU function achieves the best classification performance than others.

## 4 CONCLUSION

In the last decades, deep neural networks have acquired rapid improvements, especially in computer vision or natural language processing. Along with the layers of network getting deeper, the training efficiency and accuracy have received many concentrations which stimulates the developments of activation functions. The saturated activation functions, like Sigmoid, hyperbolic tangent, are replaced by non-saturated counterpart, such as ReLU, ELU. In this paper, the definition, pros and cons of several popular activation functions are reviewed. It should be acknowledged that some effective activation functions have not investigated in this paper, such as maxout [27], softplus [28]. The aim of this paper is to make some contributions on the understanding of the development progress, attributions, and appropriate choice of activation functions. And further investigation and analysis are required to improve the views in this paper.

## REFERENCES


[1] Y. Lecun, Y. Bengio, and G. Hinton, Deep learning, Nature, vol.521, No. 7553, 436-444, 2015.

[2] J. Schmidhuber, Deep learning in neural networks: an overview, Neural networks, vol. 61, 85-117, 2015.

[3] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, and et al, Deep learning for visual understanding: a review, Neurocomputing, vol. 187, 27-48, 2016.

[4] S. Christian, W. Liu, Y. Jia, and et al, Going deeper with convolutions, In Proceedings of the IEEE conference on computer vision and pattern recognition, 1-9, 2015.

[5] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, In Proceedings of the IEEE conference on computer vision and pattern recognition, 770-778, 2016.

[6] X. Glorot, and Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, Journal of Machine Learning Research, vol. 9, 249-256, 2010.

[7] V. Nair, and G. E. Hinton, Rectified linear units improve restricted boltzmann machines, In Proceedings of International Conference on International Conference on Machine Learning, 807-814, 2010.

[8] Y. Li, C. Fan, Y. Li, and Q. Wu, Improving deep neural network with multiple parametric exponential linear units, arXiv: 1606.00305, 2016.

[9] A. L. Maas, A. Y. Hannun, and A. Y. Ng, Rectifier nonlinearities improve neural network acoustic models, In Proceedings of International Conference on Machine Learning, Vol.30, No.1, 1-6, 2013.

[10] B. Xu, N. Wang, T. Chen, and M. Li, Empirical evaluation of rectified activations in convolutional network, arXiv:1505.00853, 2015.

[11] D. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs), arXiv:1511.07289, 2015.

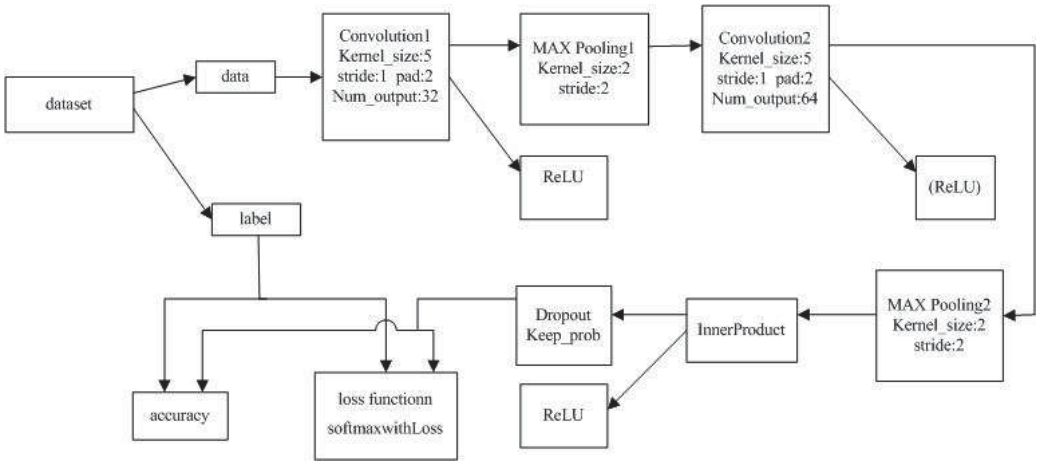

Figure 7: The structure of DCNN employed in this paper.

[12] K. He, X. Zhang, S. Ren, and J. Sun, Delving deep into rectifiers: surpassing human-level performance on ImageNet classification, In Proceedings of the IEEE international conference on computer vision, 1026-1034, 2015.

[13] C. Gulcehre, M. Moczulski, M. Denil, and Y. Bengio, Noisy activation functions, In Proceedings of International Conference on Machine Learning, 3059-3068, 2016.

[14] Y. Huang, X. Duan, S. Sun, and et al., A study of training algorithm in deep neural networks based on Sigmoid activation function, Computer measurement and control, vol. 25, no. 2, 126-129, 2017.

[15] Y. Lecun, L. Bottou, G. B. Orr, and K. Müller, Efficient Backprop, Neural networks: Tricks of the trade, Springer Berlin Heidelberg, 9-50, 1998.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, Imagenet classification with deep convolutional neural networks, In Proceedings of Advances in neural information processing systems, 1097-1105, 2012.

[17] A. L. Maas, A. Y. Hannun, and A. Y. Ng, Rectifier nonlinearities improve neural network acoustic models, In Proceedings of International conference on machine learning, Computer Science Department, vol. 30, no. 1, 2013.

[18] Q. Wei, and W. Wang, Research on image retrieval using deep convolutional neural network combining L1 regularization and PRelu activation function, In IOP Conference Series: Earth and Environmental Science , Vol. 69, No. 1, 012156, 2017.

[19] K. Simonyan, and A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv:1409.1556, 2014.

[20] M. D. Zeiler, and R. Fergus, Visualizing and understanding convolutional networks, In Proceedings of European Conference on Computer Vision, 818-833, 2014.

[21] S. Pierre, D. Eigen, X. Zhang, and et al., Overfeat: integrated recognition, localization and detection using convolutional networks, arXiv:1312.6229, 2013.

[22] K. He, X. Zhang, S. Ren, and J. Sun, Spatial pyramid pooling in deep convolutional networks for visual recognition, IEEE transactions on pattern analysis and machine intelligence, vol. 37, no. 9, 1904-1916, 2015.

[23] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, Return of the devil in the details: Delving deep into convolutional nets, arXiv:1405.3531, 2014.

[24] M. D. Zeiler, M. Ranzato, R. Monga, and et al., On rectified linear units for speech processing, In Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 3517-3521, 2013.

[25] M. Lin, Q. Chen, and S. Yan, Network in network, arXiv:1312.4400, 2013.

[26] R. K. Srivastava, J. Masci, S. Kazerounian, F. Gomez, and J. Schmidhuber, Compete to compute, In Proceedings of Advances in neural information processing systems, 2310-2318, 2013.

[27] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, Maxout networks, arXiv:1302.4389, 2013.

[28] D. Charles, Y. Bengio, F. Blisle, C. Nadeau, and R. Garcia, Incorporating second-order functional knowledge for better option pricing, In Proceedings of Advances in neural information processing systems, 472-478, 2001.

[29] P. Lennie, The cost of cortical computation, Current biology, vol. 13, no. 6, 493-497, 2003.