Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Handling Vanishing Gradient Problem Using Artificial Derivative

ZHENG HU[1], JIAOJIAO ZHANG[2], YUN GE[3]

[1]School of Software, Nanchang Hangkong University, Nanchang, China (e-mail: huzheng@nchu.edu.cn)

[2]Department of Clinical Sciences, Faculty of Medicine, Polytechnic University of Marche, Ancona, 60131, Ancona, Italy (Emails: j.zhang@pm.univpm.it)

[3]School of Software, Nanchang Hangkong University, Nanchang, China (e-mail: geyun@nchu.edu.cn)

Corresponding author: Zheng Hu (e-mail: huzheng@nchu.edu.cn).

This work was supported in part by the National Natural Science Foundation of China (41801288).

ABSTRACT Sigmoid function and ReLU are commonly used activation functions in neural networks (NN). However, sigmoid function is vulnerable to the vanishing gradient problem, while ReLU has a special vanishing gradient problem that is called dying ReLU problem. Though many studies provided methods to alleviate this problem, there has not been an efficient feasible solution. Hence, we proposed a method replacing the original derivative function with an artificial derivative in a pertinent way. Our method optimized gradients of activation functions without varying activation functions nor introducing extra layers. Our investigations demonstrated that the method can effectively alleviate the vanishing gradient problem for both ReLU and sigmoid function with few computational cost.

INDEX TERMS Activation function, Artificial derivative, ReLU, Sigmoid function, Vanishing gradient

I. INTRODUCTION

BECAUSE the nonlinear character of activation function is crucial to the universal approximative ability of neural network (NN) [1]. However, the two most used activation functions: ReLU and sigmoid [2] are probably to cause vanishing gradient, which will lead to the failure of training. The vanishing gradient problem in gradient-based training means the gradients of network weights approach zero. Therefore, it is difficult for the gradients to provide updates to the network weights. Logistic sigmoid function (the default sigmoid function in our study) and ReLU are defined as follows:

$$sigmoid(x) = \frac{1}{1+\exp(-x)} \tag{1}$$

$$ReLU(x) = \max(0, x) \tag{2}$$

Sigmoid squeezes the input from $(-\infty, +\infty)$ to the output $(0, 1)$. In deep NNs, sigmoid is likely to bring the vanishing gradient problem since the derivative is usually small. While the gradients get smaller as they go through multiple layers (with activation functions) during the backpropagation. ReLU outputs constant 0 for negative inputs, the dying ReLU problem means ReLU become inactive for the outputs and gradients are always 0, which is considered as a vanishing gradient problem as well [3].

Many methods have been proposed to reduce the vanishing gradient problem. A group of methods modify the activation functions to alleviate the problem, for example, there are several variants for ReLU [4] [5] [6] [7] [8], and variants for sigmoid [9] [10]. The variants commonly need more computational cost, and their performance varies on different tasks and data sets [2]. Some methods introduce extra layers into NNs, like normalization techniques [11] [12] [13]. Though their effectiveness, they will tremendously increase the computational cost [14]. While the methods only focusing on the initialization of neural network weights and biases [15] [6] [14] [3] can hardly require extra computational cost [3]. But they cannot effectively solve the vanishing gradient problem.

In this study, we use a novelty method to deal with the vanishing gradient of ReLU and sigmoid, which replaces the original derivatives with an artificial derivative (AD). This idea was firstly used in previous studies [16] [17], when they facing problems that real derivatives were infeasible. In current study, we started to focus on ADs, and applied them to activation functions for better performance. Instead of changing the activation functions, the ADs only modifies their derivatives. To avoid the vanishing gradient, the main idea of this solution is properly amplifying derivative functions of activation functions by replacing them with AD. Generally, the AD should be consistent with the original one while a significant and limited amplification is applied.

In sum, there are three types of methods:1) modeling methods such as modifying activation functions or creating extra layers; 2) parameter methods such as initialization optimization; 3) training solutions such as applying ADs. The ADs will only affect the computational cost of training. If an AD required too much computational cost, we could periodically return it to the original derivative to cut down the computational overhead. Though AD is usually computational cheap, it is very effective for training optimization. ADs make a smaller change to the NN than modeling solutions. ADs are much more effective than initialization solutions. Our method has been demonstrated more effective than Leaky ReLU (LReLU) [5] that is considered as a typical substitute for ReLU. The Leaky ReLU and its derivative defined as follows:

$$f(x) = \max(\alpha \cdot x, x) \quad \alpha = 0.01 \tag{3}$$

$$f'(x) = \begin{cases} \alpha & x < 0 \\ 1 & x \geqslant 0 \end{cases} \quad \alpha = 0.01 \tag{4}$$

Improved logistic sigmoid (Isigmoid) is a variant for sigmoid, which has a similar improvement idea to LReLU [9]. The Isigmoid and its derivative are defined as follows:

$$f(x) = \begin{cases} \alpha \cdot (x - a) + sigmoid(a) & x \geqslant a \\ sigmoid(x) & -a < x < a \\ \alpha \cdot (x + a) + sigmoid(-a) & x \leqslant -a \end{cases} \tag{5}$$

$$f'(x) = \begin{cases} \alpha & |x| \geqslant a \\ sigmoid'(x) & |x| < a \end{cases} \tag{6}$$

Its parameters were originally set to $a = 0.01$ and $\alpha = 7$ [9]. To avoid the gradient exploding problem, the parameters were set to $a = 0.01$ and $\alpha = 1$ in this study.

II. ARTIFICIAL DERIVATIVES

Derivatives are fundamental to the gradients in gradient-based learning. There are many approaches to optimize gradients. However, almost no derivative is involved. In this study, we attempted to replace the derivative with an artificial substitute. Considering the original derivatives indicate the actual changes of function, AD must not play the opposite role to them. Additionally, AD must not be too large, otherwise, it tends to cause gradient exploding problems. Presently, AD was a heuristic method, and there was some arbitrariness in constructing an AD as well. In this study, we proposed two ADs for ReLU and sigmoid which could amplify original derivatives' parts equal to 0 and close to 0, thus issue their vanishing gradient problems effectively.

A. ARTIFICIAL DERIVATIVE FOR RELU

The AD was applied to ReLU was logistic function as a smooth approximation of ReLU's original derivative. Generally, logistic function is also consistent with the original derivative, which will not make training collapsed. The original derivative (d_0) and the AD (d_1) are as follows:

$$d_0(x) = \begin{cases} 0, x < 0 \\ 1, x \geq 0 \end{cases} \tag{7}$$

$$d_1(x) = \frac{1}{1 + \exp(-x)} \tag{8}$$

Compared with the original derivative that only outputs zero for negative inputs, the AD output small positive values for negative inputs. Since all the values of the AD are positive, it has a bigger possibility to avoid dying ReLU than original one. The comparison of these two derivatives is shown in Fig. 1.

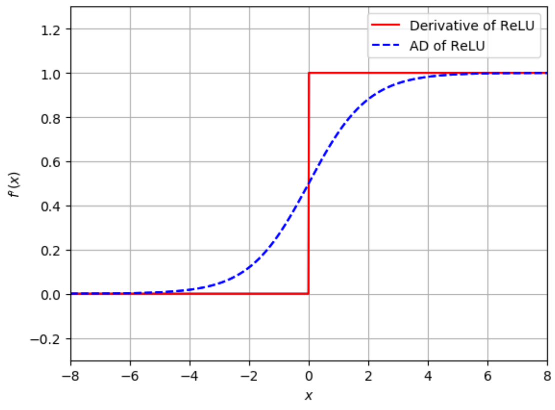

FIGURE 1. Comparison of ReLU's derivative and AD

B. ARTIFICIAL DERIVATIVE FOR SIGMOID

The derivative of Sigmoid is bell-shaped, which approaches zero as the input approaches $\pm\infty$. Logistic sigmoid's original derivative (d_0) can be computed based on its output y (which has been computed in "forward propagation"):

$$y = \frac{1}{1 + \exp(-x)} \tag{9}$$

$$d_0(x) = y \cdot (1 - y) \tag{10}$$

AD which can amplify original derivative d_0 was proposed as follows:

$$d_1(x) = \min(y, 1 - y) \tag{11}$$

Comparing with original derivative, this modified AD is amplified and simplified as well. Despite the obvious amplification, generally it is still consistent with original derivative also small. The comparison of the original derivative and this AD are shown in Fig. 2.

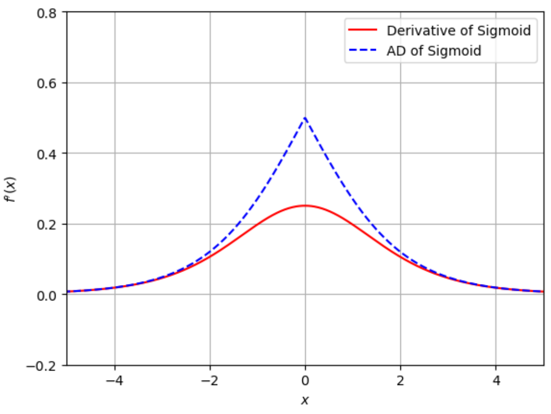

FIGURE 2. Comparison of sigmoid's derivative and AD

III. EXPERIMENTS

To verify AD's advantage over the original one in vanishing gradient problems, deep NNs with vanishing gradient problem should be used. However, such problematical NNs can hardly be published. Therefore, we applied the derivatives to particular NNs in a special kind of approximation experiment. Considering these neural networks and tests were deliberately designed to solve vanishing gradient problems, a typical NN and an image classification test were also applied. Furthermore, we roughly extended this typical NN to create a vanishing gradient problem. Thus, the abilities to deal with the vanishing gradient problem were evaluated.

A. APPROXIMATION EXPERIMENT WITH RELUS

Recently, Lu Lu and Yeonjong Shin proposed a reasonable approximation test to evaluate solutions for the dying ReLU problem [3]. In this study, we did a similar experiment on ReLUs. In this experiment, we attempted to approximate $f(x) = |x|$ by using n-layer ReLU networks with hidden layers of width=2. The training data were 3000 float numbers randomly chosen from $\left(-\sqrt{3}, \sqrt{3}\right)$, the test data were 1000 float numbers randomly chosen from $\left(-\sqrt{3}, \sqrt{3}\right)$. In addition, the trainable weights and biases were 32 bits float number parameter, the n-layer ReLU network contained $4 + 4 \cdot (n-2)$ weights and $3 + 2 \cdot (n-2)$ biases.

Although 2-layer ReLU network is enough to approximate ($|x| = ReLU(x) + ReLU(-x)$), deeper ReLU networks were significant in evaluating dying ReLU. Herein we tested original ReLU, Leaky ReLU (α is 0.01), and AD ReLU (the AD is $sigmoid(3.5 \cdot x)$) with n-layer ReLU networks, a n-layer ReLU network has $n-1$ ReLUs. Although Leaky ReLU is different from ReLU without dying ReLU problem, we did this comparison to observe how much ReLU can be optimized by the AD. To demonstrate the significance of AD, all the trainings were evaluated under two different sets of hyperparameters that were marked as hyperparameter set A and hyperparameter set B. The hyperparameter sets are elaborated in Table 1:

TABLE 1. Details of hyperparameter set A and set B

	Set A	Set B
Network width	2	2
Network layers	2,4,6,8	2,4,6,8
Training epochs	300	300
Mini-batch size	50	100
Weights initialization	Common initialization	He initialization
Optimizer	Adam	SGD
Loss function	MSE	MSE

In the above hyperparameters, common initialization is the default initialization of PyTorch (the deep learning framework used in this study). The initialization obeys a uniform distribution:

$$\mathcal{U}(-k, k), k = \sqrt{\frac{1}{in\_features}} \tag{12}$$

The other initialization (He initialization) is proposed for deep convolutional neural networks (CNNs) with ReLU [6]. It obeys a normal distribution:

$$\mathcal{N}(0, std), std = \sqrt{\frac{2}{in\_features}} \tag{13}$$

In optimizers, adaptive moment estimation (Adam) is an adaptive learning rate and momentum optimization algorithm that was been proposed specifically for training deep NNs [18]. Stochastic gradient descent (SGD) is the original optimization approach used for gradient-based learning. A simple loss function, mean square error (MSE), was applied in both hyperparameter sets.

AD ReLU requires more computational cost, AD and original derivative can be converted to each other in training to reduce the computational cost. We set the conversion condition to training loss of last epoch exceeds A times of best loss (we set $A = 1.035$).

In this approximation test, losses with different activations and hyperparameters were compared to evaluate the performances of AD and original derivative. Each loss was the best (lowest) loss in a single training, each kind of training was performed 100 times independently. Instead of listing all the losses, we counted the losses in different intervals, in addition, the average losses of same kind of trainings were calculated. The experiment results are shown in Table 2, Fig. 3, and Fig. 4.

In the experiments, many losses were around some local minimums due to the vanishing gradient problem, and the intervals in Fig. 3 and Fig. 4 are based on the distribution of the losses.

The losses of NNs' increase with the number of layers. Losses of original ReLU NNs with hyperparameter set A (Fig. 3) and set B (Fig. 4) are similar, more than 80% of losses are over 0.2 for the 8-layer network with both hyperparameter set A and set B. LReLU can alleviate vanishing gradient effectively. LReLU has better performance with hyperparameter set A than set B, it has most of the losses less than 0.001 with set A. However, its performance is not good in

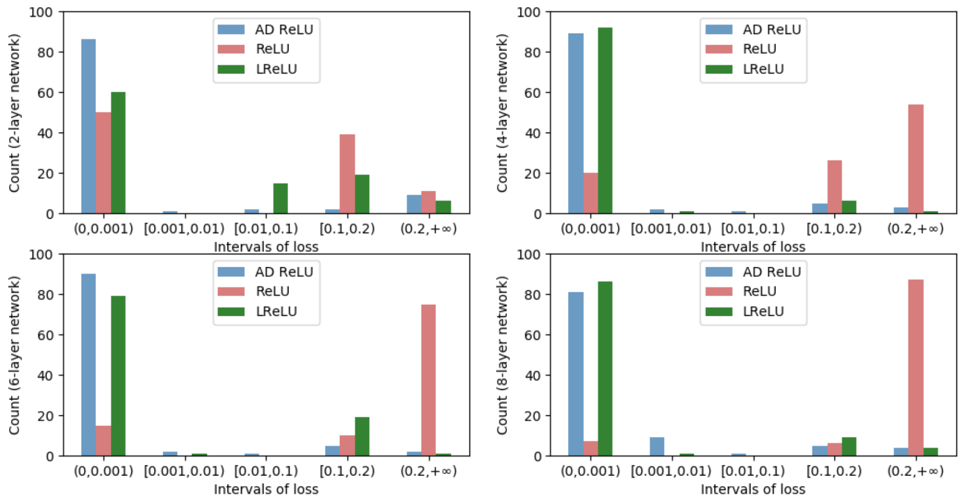

FIGURE 3. Testing losses with AD ReLU, original ReLU and Leaky ReLU under hyperparameter set A

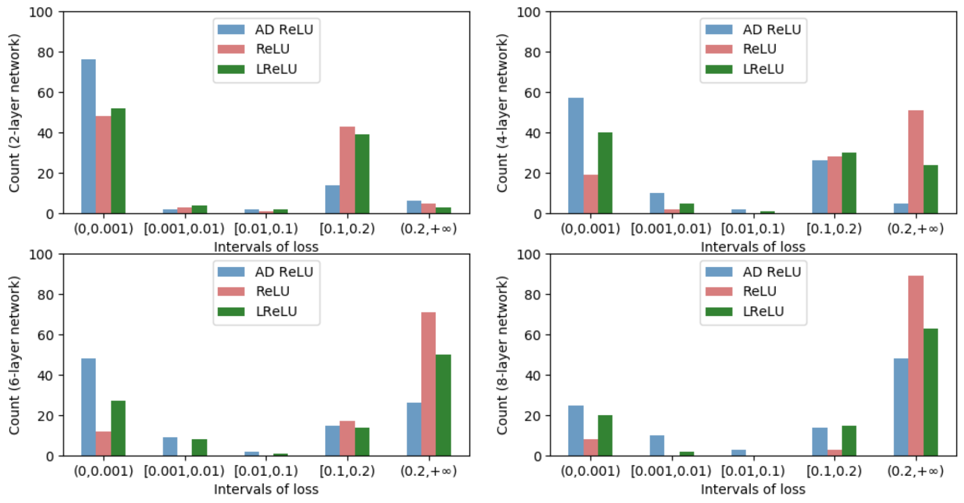

FIGURE 4. Testing losses with AD ReLU, original ReLU and Leaky ReLU under hyperparameter set B

TABLE 2. Average testing losses with AD ReLU, original ReLU and Leaky ReLU

Parameter set A			
Layers	AD ReLU	ReLU	Leaky ReLU
2	0.026	0.089	0.053
4	0.016	0.171	0.013
6	0.014	0.195	0.034
8	0.019	0.217	0.025
Parameter set B			
Layers	AD ReLU	ReLU	Leaky ReLU
2	0.0387	0.0789	0.0711
4	0.0557	0.161	0.105
6	0.0871	0.189	0.138
8	0.136	0.209	0.169

the 2-layer network. AD ReLU can also alleviate vanishing gradient like LReLU, with better performance than LReLU in the 2-layer network. AD ReLU also has better performance with hyperparameter set A than set B. Except for the 2-layer network, LReLU and AD ReLU's performances are about the same with hyperparameter set A, and AD ReLU's performances are significantly better than LReLU with hyperparameter set B.

In conclusion, the original ReLU always had a serious vanishing gradient problem in this approximation task. Although LReLU can alleviate vanishing gradient effectively, its performances are not optimal in the 2-layer network for its waveform not being suitable as ReLU nor AD ReLU in this approximation task. The main cause of the vanishing gradient problem for ReLU is not its waveform. Applying the AD for ReLU can alleviate the problem effectively, and even can achieve better performance than using the variant of ReLU.

B. APPROXIMATION EXPERIMENT WITH SIGMOIDS

To evaluate the effectiveness of the AD for sigmoid, we did an analogous experiment with n-layer sigmoid networks (an n-layer sigmoid network contains n logistic sigmoid functions). The test is simple two numbers (v_1, v_2) comparison, while the target will be 1 if $v_1 < v_2$, and be 0 otherwise. Sigmoid networks with hidden layers of width=2 were tested in this experiment. The training data were 3000 float numbers randomly chosen from $(-\sqrt{3}, \sqrt{3})$, the test data were 500 float numbers randomly chosen from $(-\sqrt{3}, \sqrt{3})$. In addition, the trainable weights and biases were 32 bits float number parameters, the n-layer sigmoid network contained $6 + 4 \cdot (n-2)$ weights and $3 + 2 \cdot (n-2)$ biases.

In this experiment, we tested the original sigmoid, Isigmoid, and AD sigmoid with n-layer NNs. Although Isigmoid is different from sigmoid, and literally has no vanishing gradient problem. We did their comparisons to see how much sigmoid can be optimized by the AD. The training setup was similar to ReLU networks, and the hyperparameter sets are elaborated in Table 3.

TABLE 3. Details of hyperparameter set A and set B

	Set A	Set B
Network width	2	2
Network layers	12,13,14,15	4,5,6,7
Training epochs	300	300
Mini-batch size	50	100
Weights initialization	Common initialization	Xavier initialization
Optimizer	Adam	SGD
Loss function	MSE	MSE

In these hyperparameters, layers in set A were deeper than in set B. The reason was Adam being much more effective than SGD in handling the vanishing gradient. Although Xavier initialization is preferable for sigmoid [15], the 7-layer sigmoid networks could not be trained well under hyperparameter set B. We applied the common and Xavier initialization in the NNs, Xavier initialization obeys a normal distribution:

$$\mathcal{N}(0, std), std = \sqrt{\frac{2}{in\_features + out\_features}} \quad (14)$$

In this approximation test, losses with different activations and hyperparameters were compared to evaluate the performances of AD and original derivative. Like the approximation test for ReLUs, the testing losses were detailed record and calculated. The experiment results are shown in Table 4, Fig. 5, and Fig. 6.

In the experiments, many losses were around certain local minimums due to the vanishing gradient problem, and the intervals in Fig. 5 and Fig. 6 are based on the distribution of the losses.

TABLE 4. Average testing losses with AD sigmoid, original sigmoid and Isigmoid

Parameter set A			
Layers	AD sigmoid	sigmoid	Isigmoid
12	0.012	0.068	0.056
13	0.019	0.169	0.052
14	0.019	0.221	0.053
15	0.029	0.249	0.053
Parameter set B			
Layers	AD sigmoid	sigmoid	Isigmoid
4	0.044	0.217	0.075
5	0.149	0.249	0.241
6	0.233	0.249	0.937
7	0.248	0.249	0.62

For sigmoid and AD sigmoid, the losses of NNs increase to a high level with the number of layers, and Isigmoid networks are not sensitive to the number of layers. For all three types of neural networks, the performances with hyperparameter set A (Fig. 5) are better than set B (Fig. 6). In the 15-layer original sigmoid network with hyperparameter set A, and 5,6,7-layer original sigmoid network with set B, nearly all the losses are more than 0.2. Because Isigmoid's output range $(-\infty, +\infty)$ is very different from sigmoid, it will not have the vanishing gradient problem. Isigmoid's performances change slightly with the number of layers, and all its losses are relatively good. However, only a small part of Isigmoid's losses is the best level. AD sigmoid also has better performances than original sigmoid, it is better than Isigmoid with hyperparameter set A, and not so good as Isigmoid in 6,7-layer networks with hyperparameter set B.

The results exhibited the following conclusions: original sigmoid always had a serious vanishing gradient problem in this approximation task. Although Isigmoid can avoid most vanishing gradient problems, it is hard to achieve the best level, because its waveform not as suitable as sigmoid or AD sigmoid in this approximation task. Applying the AD for sigmoid can alleviate the problem effectively, and can even achieve better performance than Isigmoid as well.

C. EXPERIMENT WITH ARTIFICIAL DERIVATIVES ON CIFAR-100 TASK

In addition, we compared the performances of the original derivatives and ADs on the CIFAR-100 dataset [19] with AlexNet [20]. CIFAR-100 dataset has 100 classes of images (the initial accuracy is 1%), each class containing 600 images (500 training images and 100 testing images). ReLU, AD ReLU, sigmoid and AD sigmoid were applied to AlexNet. The total number of AlexNet's parameters was 2495396. The hyperparameters with ReLUs and sigmoids were shown in Table 5.

Cross entropy loss is a commonly used loss function for CIFAR-100 test. In AlexNet, using ReLU will produce better performance than sigmoid [20]. The average performances over 5 independent training were shown in Fig. 7 and Fig. 8.

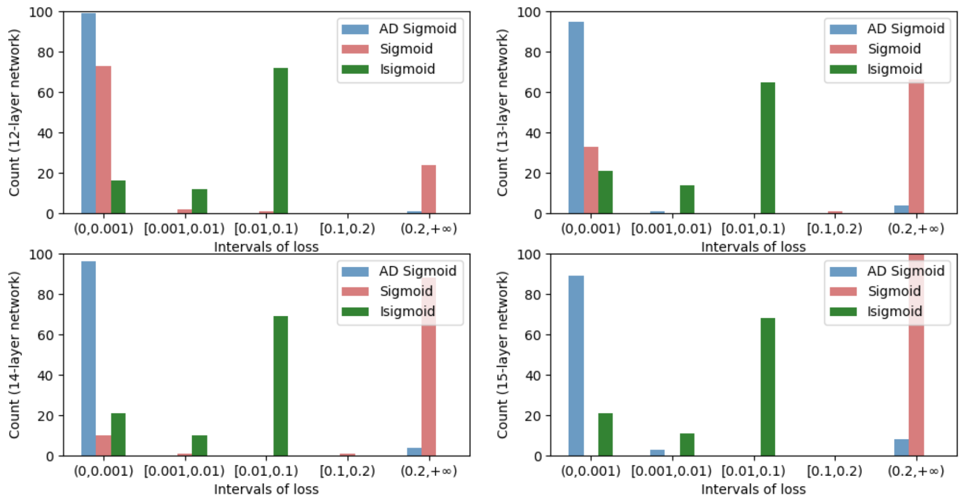

FIGURE 5. Testing losses with AD sigmoid, original sigmoid and Isigmoid under hyperparameter set A

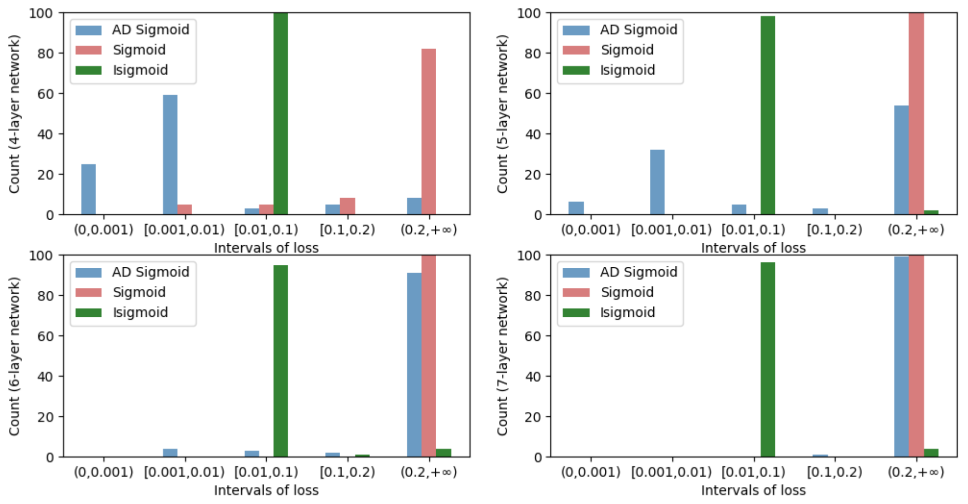

FIGURE 6. Testing losses with AD sigmoid, original sigmoid and Isigmoid under hyperparameter set B

TABLE 5. Details of hyperparameters for ReLUs and sigmoids

	ReLU, AD ReLU	sigmoid, AD sigmoid
Training epochs	300	300
Mini-batch size	200	200
Weights initialization	He initialization	Xavier initialization
Optimizer	SGD	Adam
Loss function	Cross entropy loss	Cross entropy loss

Since AlexNet is well designed, and vanishing gradient problem is not likely to happen on it. Applying AD can only contribute few promotions. However, it still showed the AD was not only effective in simple and extreme networks (in the approximation tasks) but also useful in this common task.

AlexNet is well designed to avoid vanishing gradient problems, we roughly extended the layers of AlexNet (from 5 convolutional layers to 13 convolutional layers) to make the vanishing gradient problem more likely to happen. The total number of parameters in extended AlexNet was 7488292, it is about three times the parameter amount of AlexNet. The comparison of original AlexNet and extended AlexNet was shown in Table 6.

Then we did a similar CIFAR-100 experiment with ReLU and AD ReLU. The experiment result was shown in Fig. 9. It shows original ReLU failed all the trainings, but the AD

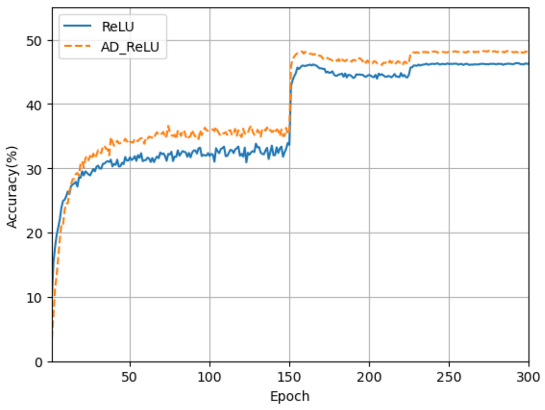

FIGURE 7. Comparison of AlexNet with ReLU and with AD ReLU in CIFAR-100 task

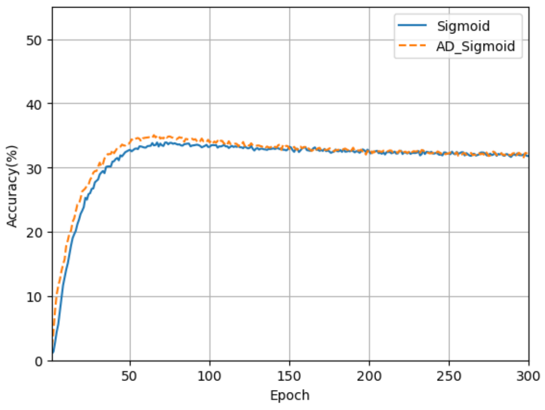

FIGURE 8. Comparison of AlexNet with sigmoid and with AD sigmoid in CIFAR-100 task

TABLE 6. Original AlexNet and extended AlexNet

Original AlexNet	Extended AlexNet
Conv(3,64)	Conv(3,48)
MaxPool	Conv(48,64)
Conv(64,192)	Conv(64,96)
MaxPool	Conv(96,128)
Conv(192,384)	Conv(128,160)
Conv(384,256)	MaxPool
Conv(256,256)	Conv(160,224)
MaxPool	Conv(224,288)
FC	Conv(288,352)
	MaxPool
	Conv(352,416)
	Conv(416,352)
	Conv(352,288)
	Conv(288,224)
	Conv(224,224)
	MaxPool
	FC

ReLU is still able to train this deep network.

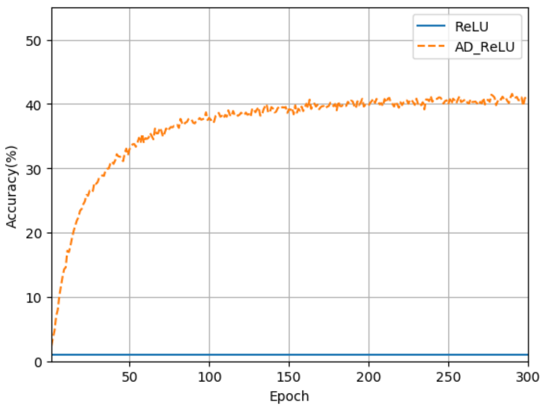

FIGURE 9. Comparison of extended AlexNet with ReLU and with AD ReLU in CIFAR-100 task

IV. CONCLUSION

During the backpropagation, activation functions' derivatives are pivots for network weights updating. If the original derivative functions had defects with backpropagation, applying a manual refinement (artificial derivative/ AD) is possible. The AD not only has stronger advantage in handling vanishing gradient problems than original derivative, but also requires fewer computational costs in most situations. In this work, two ADs were designed and applied to ReLU and sigmoid. Compared with original derivatives, the ADs can provide better performance in our experiments. AD is a generic method that can be applied to any activations including their variants (i.e., LReLU and Isigmoid). Additionally, because the ADs only affect the training, the trained model will not bring in any extra computational cost.

REFERENCES

[1] A. Heinecke, J. Ho, and W.-L. Hwang, "Refinement and universal approximation via sparsely connected relu convolution nets," IEEE Signal Processing Letters, vol. 27, pp. 1175–1179, 2020.
[2] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," arXiv preprint arXiv:1710.05941, 2017.
[3] L. Lu, Y. Shin, Y. Su, and G. E. Karniadakis, "Dying relu and initialization: Theory and numerical examples," arXiv preprint arXiv:1903.06733, 2019.
[4] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in Proceedings of the 27th international conference on machine learning (ICML-10), 2010, pp. 807–814.
[5] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in Proc. icml, vol. 30, no. 1, 2013, p. 3.
[6] 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, 2015, pp. 1026–1034.
[7] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," arXiv preprint arXiv:1511.07289, 2015.
[8] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in Advances in neural information processing systems, 2017, pp. 971–980.

[9] Y. Qin, X. Wang, and J. Zou, "The optimized deep belief networks with improved logistic sigmoid units and their application in fault diagnosis for planetary gearboxes of wind turbines," IEEE Transactions on Industrial Electronics, vol. 66, no. 5, pp. 3814–3824, 2018.
[10] X. Wang, Y. Qin, Y. Wang, S. Xiang, and H. Chen, "Reltanh: An activation function with vanishing gradient resistance for sae-based dnns and its application to rotating machinery fault diagnosis," Neurocomputing, vol. 363, pp. 88–98, 2019.
[11] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," arXiv preprint arXiv:1502.03167, 2015.
[12] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," arXiv preprint arXiv:1607.06450, 2016.
[13] Y. Wu and K. He, "Group normalization," in Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 3–19.
[14] D. Mishkin and J. Matas, "All you need is a good init," arXiv preprint arXiv:1511.06422, 2015.
[15] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in Proceedings of the thirteenth international conference on artificial intelligence and statistics, 2010, pp. 249–256.
[16] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," arXiv preprint arXiv:1308.3432, 2013.
[17] S. Shalev-Shwartz, O. Shamir, and S. Shammah, "Failures of gradient-based deep learning," in Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017, pp. 3067–3075.
[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.
[19] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.
[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in neural information processing systems, 2012, pp. 1097–1105.

ZHENG HU was born in 1986. He received the B.S. degrees in electronic engineering from the Nanchang University, Charlottesville, in 2008 and the Ph.D. degree from University of Chinese Academy of Sciences, in 2014. Since 2015, he was a lecturer in Nanchang Hangkong University. He is the author of 6 articles. His research interests include neural network, machine learning and natural language processing.

JIAOJIAO ZHANG , Ph. D in Food Science of Zhejiang University, Doctor Candidate in Biomedical Science of Polytechnic University of Marche, Italy. During doctoral period, her research filed involved the analysis of active substances in natural products (pharmacological and medicinal chemistry) and food nutrition. Research contents also include the immunotherapy of melanoma, the appliances of nanoparticle carriers and artificial intelligence of melanoma diagnosis.

YUN GE was born in 1983. She received the B.S. degree in Computer Science and Technology from HeFei University of Technology, Hefei, China, in 2005, and the M.S. degree in Computer Science from Central Sourth University, Hunan, China, in 2009. She received the Ph.D. degree at the School of Information Engineering, Nanchang University, Nanchang ,China, in 2019. She is the author of 20 articles. Her current research interests include machine learning and remote sensing image retrieval.

• • •