Adaptive piece-wise linear squashing activation function for deep neural networks

Yevgeniy Bodyanskiy[1], Nonna Kulishova[1], Mykhailo Petrykin[1] and Maxim Kulishov[1]

[1] *Kharkiv National University of Radio Electronics, 14 Nauki av., Kharkiv, 61166, Ukraine*

Abstract

The work considers the vanishing gradient problem inherent in deep neural networks and limiting their potential due to the unstable learning process. An adaptive piecewise linear squashing activation function (APWLSAF) is proposed. This function, on the one hand, will ensure high accuracy of the deep network, and on the other hand, will make the network learning process stable due to the fact that the gradient of proposed function does not vanish and, therefore, will not be able to stop the adjusting network parameters.

Keywords

Activation function, deep neural network, classification, image recognition

1. Introduction

For shallow (SNN) and deep neural networks (DNN), a wide range of activation functions are used. Typically, such functions must guarantee the ability to discriminate between small and large input signals, as well as provide fast and reliably stable network training. In recent years, many activation functions have been proposed that best meet these requirements [1]. They can be divided into several main groups depending on the type of curve, properties of monotonicity, smoothness, continuity, etc. Adaptive and parametric activation functions are of greatest interest, since in various works [2 - 6] it was found that the choice of type and parameters of activation function seriously determine the accuracy of artificial neural networks, regardless of their architecture and application tasks.

In this regard, parametric adaptive functions have undoubted advantages, since they provide the ability to customize their form in accordance with a specific architecture and task. The most famous among adaptive activation functions are the Parametric Rectification Linear Unit PReLU [7], and a whole family of similar functions: Adaptive Piecewise Linear function APL [8], Adaptive Activation Function AAF [9], S-shaped ReLU [10], Multi-bin Trainable Linear Unit MTLU [11], Swish [12].

The adaptability of the mentioned and other activation functions can be controlled thanks to their piecewise structure. Most often, such functions are a construction where fragments of both linear and nonlinear functions can be connected in series [13, 14, 1]: ABReLU, AdPReLU CELU, CReLU, ELU, LReLU, MeLU, PDELU, PELU, pTanh, PTELU, RePU, SELU, S.L.U. The goal of each such structure is creation of a certain curve shape to describe the dependence of neuron output parameters from input ones. Among the proposed activation functions there are sigmoidal, linear, and exponential dependencies. In shallow and deep neural networks, different activation functions are used for different types of network layers (input, output, intermediate) and they are selected manually or by default.

ProfIT AI 2024: 4th International Workshop of IT-professionals on Artificial Intelligence (ProfIT AI 2023), September 25–27, 2024, Cambridge, MA, USA

yevgeniy.bodyanskiy@nure.ua (Ye. Bodyanskiy); nonna.kulishova@nure.ua (N. Kulishova); mykhailo.petrykin@nure.ua (M. Petrykin); absolutezero@i.ua (M. Kulishov)

0000-0001-5418-2143 (Ye. Bodyanskiy); 0000-0003-1142-4100 (N. Kulishova); 0009-0006-5983-8950 (M. Petrykin); 0009-0003-2867-2132 (M. Kulishov)

The sheer number of existing adaptive activation functions suggests that in each task, researchers choose the type of function based on subjective ideas about how well it will provide the desired level of result. If the function was not chosen very well, then adjusting its parameters during network training will only slightly improve the situation, but not radically change it. For many tasks, especially those related to big data, processing online data streams, etc. it is not always possible to select the most appropriate activation functions type for individual layers of a deep network during a sequence of experiments, which ultimately makes the result quality dependent on the developer experience and intuition. Thus, it is necessary to construct an activation function that will allow a complete change in its form during network training, and will ensure high accuracy and speed of learning.

This work proposes a parametric adaptive piecewise linear activation function, which can change shape from sigmoidal to linear, and an algorithm for its training.

2. Piece-wise linear squashing activation function

Modern commercial applications designed, for example, for classification, object recognition in video, and performance prediction, are based on use of shallow and deep feedforward networks. Their architectures can be considered as multilayer perceptrons, which have been known for a long time [15]. The nature of transformation performed by a neural network as a model depends on activation function of neurons in its composition.

Among the squashing activation functions used in traditional shallow and deep neural networks, and satisfy the conditions of Cybenko's approximation theorem [16], the hyperbolic tangent function has become the most widespread

$$\hat{y}_j(k) = \psi_j\left(\theta_{j0} + \sum_{i=1}^{n} w_{ij}x_i(k)\right) = \psi_j\left(\sum_{i=0}^{n} w_{ij}x_i(k)\right) = \\ = \psi_j\left(w_j^T x(k)\right) = \psi_j\left(u_j(k)\right) = \tanh\left(\gamma_j u_j(k)\right) = \frac{1 - e^{-2\gamma_j u_j(k)}}{1 + e^{-2\gamma_j u_j(k)}} \quad (1)$$

where $\hat{y}_j(k)$ - j-th SNN neuron output signal at the moment of discrete time k = 1, 2,..., n, $\psi_j(\cdot)$- nonlinear activation function of this neuron, θ_{j0} - bias term, n is the number of inputs to j-th neuron, w_{ij}- tuned synaptic weight on i-th input of j-th neuron, $x_i(k)$ - input signal on i-th input of j-th neuron at k-th moment of current time, $\theta_{j0} \equiv w_{j0}, x(k) = \left(1, x_1(k), \ldots, x_i(k), \ldots, x_n(k)\right)^T$ - $(n+1) \times 1$ - input vector, $w_j = \left(w_{j0}, w_{j1}, \ldots, w_{ji}, \ldots, w_{jn}\right)^T$- $(n+1) \times 1$ - vector of adjustable synaptic weights, $u_j(k)$ - a signal of internal activation of j-th neuron, γ_j - gain parameter, which determines the form of this activation function.

Derivative of this feature used in gradient tuning process of neuron looks like

$$\psi'\left(u_j\right) = \frac{\partial\psi(u_j)}{\partial u_j} = \gamma_j\left(1 - \left(Tanh\left(\gamma_j u_j\right)\right)^2\right) = \gamma_j\left(1 - \hat{y}_j^2\right), \quad (2)$$

from where it follows that when output signal approach the values ±1 the derivative goes to zero, that is, learning process stops due to the so-called "vanishing gradient" effect.

It is interesting to see that to Cybenko theorem conditions also corresponds the adaptive-linear function Satlin

$$\psi_j\left(u_j(k)\right) = Satlin\left(u_j(k)\right) = \begin{cases} -1, & u_j(k) < -1 \\ u_j(k), & -1 \le u_j(k) \le 1 \\ 1, & u_j(k) > 1 \end{cases} \quad (3)$$

the derivatives of which take zero value when output of internal activation signal goes out of bounds of interval [-1, 1]. Therefore, Satlin is not used at all in gradient learning of neural networks.

It is the vanishing gradient effect that has led to the fact that in DNN squashing activation functions are not used at all, and the most widespread are piece-wise linear functions of ReLU type:

$$\psi_j\left(u_j(k)\right) = \mathrm{Re\,L\,U}\left(u_j(k)\right) = max\left(0, u_j(k)\right) = \begin{cases} u_j(k), u_j(k) \geq 0 \\ 0, \quad otherwise \end{cases} \tag{4}$$

or PreLU:

$$\psi_j\left(u_j(k)\right) = \mathrm{PRe\,L\,U}\left(u_j(k)\right) = max\left(0, u_j(k)\right) = = \begin{cases} u_j(k), & u_j(k) \geq 0 \\ au_j(k), 0 < a < 1, & otherwise \end{cases} \tag{5}$$

The advantage of these functions is simplicity of their derivative, which facilitates the learning process.

Since these functions do not satisfy approximation theorem conditions, number of neurons and layers is significantly increased to ensure the required quality in DNN, which leads, first, to a significant reduction of learning rate, and secondly, it requires an increase training data number which are not always available when solving real practical problems.

Therefore, it is advisable to introduce an activation piece-wise linear function, which will have simple derivative, is close enough to squashing functions with their approximation properties, and does not suffer from the effect of "vanishing gradient".

As such a function is offered a formulation of Adaptive Piece-Wise Linear Squashing Activation Function (APWLSAF)

$$\psi_j\left(u_j(k)\right) = \begin{cases} u_j(k), & \text{-1} \leq u_j(k) \leq 1 \\ 1 - a_j^R\left(1 - u_j(k)\right), & u_j(k) > 1 \\ a_j^L\left(1 + u_j(k)\right) - 1, & u_j(k) < -1 \end{cases}. \tag{6}$$

Its graph is shown in Figure 1.

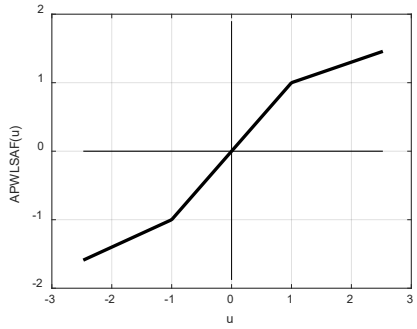


Figure 1: Adaptive Piece-Wise Linear Squashing Activation Function (APWLSAF)

It is easy to see that when $a_{0j}^R = a_{0j}^L = 0$ APWLSAF is converted to Satlin, and when $a_j^R = a_j^L = 0$ we get an elementary linear function ALA (Adaptive Linear Associator). This activation function is characterized by the simplicity of its derivatives:

$$\frac{\partial\psi_j(u_j)}{\partial u_j} = \begin{cases} 1, \text{-1} \leq u_j(k) \leq 1 \\ a_j^R, \; u_j(k) > 1 \\ a_j^L, \; u_j(k) < -1 \end{cases}, \qquad \begin{cases} \frac{\partial\psi_j(u_j)}{\partial a_j^R} = 1 - u_j, \\ \frac{\partial\psi_j(u_j)}{\partial a_j^L} = 1 + u_j \end{cases} \tag{7}$$

and, if you set additional limitations $a_j^R \geq \varepsilon$, $a_j^L \geq \varepsilon$, it is protected from the effect of the "vanishing gradient" inherent in Satlin.

APWLSAF-based neural network approximation properties can be improved if we not only set up synaptic weights w_{ij} in the learning process, but also tune the parameters of activation functions a_j^R and a_j^L.

3. Adaptive neuron parameters training with APWLSAF

The process of adjusting each neuron in network is implemented by minimizing the accepted criterion of learning $E_j(k)$, most often quadratic, with the help of so-called δ-rule, which is essentially a procedure of gradient optimization $E_j(k)$ on tuned synaptic weights w_{ij}.

Therefore, if training criterion is used as

$$E_j(k) = \frac{1}{2}\left(y_j(k) - \hat{y}_j(k)\right)^2 = \frac{1}{2}e_j^2(k) = \frac{1}{2}\left(y_j(k) - \psi_j\left(u_j(k)\right)\right)^2 = \\ = \frac{1}{2}\left(y_j(k) - \psi_j\left(\sum_{i=1}^{n} w_{ij}x_i(k)\right)\right)^2 = \frac{1}{2}\left(y_j(k) - \psi_j\left(w_j^T x(k)\right)\right)^2 \quad (8)$$

(here $y_j(k)$- external learning signal), then the procedure for optimizing learning will look like:

$$w_{ij}(k) = w_{ij}(k-1) - \eta(k)\frac{\partial E_j(k)}{\partial e_j(k)}\frac{\partial e_j(k)}{\partial w_{ij}} = w_{ij}(k-1) - \eta(k)e_j(k)\frac{\partial e_j(k)}{\partial w_{ij}} = \\ = w_{ij}(k-1) - \eta(k)e_j(k)\frac{\partial e_j(k)}{\partial u_j(k)}\frac{\partial u_j(k)}{\partial w_{ij}} = w_{ij}(k-1) + \eta(k)e_j(k)\psi_j'\left(u_j(k)\right)x_i(k) = \\ = w_{ij}(k-1) + \eta(k)\delta_j(k)x(k) \quad (9)$$

or in a vector form:

$$\boldsymbol{w}_j(k) = \boldsymbol{w}_j(k-1) + \boldsymbol{\eta}(k)\boldsymbol{\delta}_j(k)\boldsymbol{x}(k) \quad (10)$$

where $\boldsymbol{\delta}_j(k)$ - the so-called δ-error; $\boldsymbol{\eta}(k)$ is a search step parameter that is selected from one or another consideration.

For our case, the learning algorithm can be recorded in a sufficiently simpler form:

$$\boldsymbol{w}_j(k) = \begin{cases} \boldsymbol{w}_j(k-1) + \boldsymbol{\eta}(k)e_j(k)\boldsymbol{x}(k), & -1 \le \mathrm{u}_j(k) \le 1 \\ \boldsymbol{w}_j(k-1) + \boldsymbol{\eta}(k)e_j(k)a_j^R\boldsymbol{x}(k), & \mathrm{u}_j(k) > 1 \\ \boldsymbol{w}_j(k-1) + \boldsymbol{\eta}(k)e_j(k)a_j^L\boldsymbol{x}(k), & \mathrm{u}_j(k) < -1 \end{cases} \quad (11)$$

or

$$\boldsymbol{w}_j(k) = \begin{cases} \boldsymbol{w}_j(k-1) + \boldsymbol{\eta}(k)e_j(k)\boldsymbol{x}(k), & -1 \le \mathrm{u}_j(k) \le 1 \\ \boldsymbol{w}_j(k-1) + \boldsymbol{\eta}(k)e_j(k)\boldsymbol{x}^R(k), & \mathrm{u}_j(k) > 1 \\ \boldsymbol{w}_j(k-1) + \boldsymbol{\eta}(k)e_j(k)\boldsymbol{x}^L(k), & \mathrm{u}_j(k) < -1 \end{cases} \quad (12)$$

where $\boldsymbol{x}^R(k) = a_j^R\boldsymbol{x}(k), \boldsymbol{x}^L(k) = a_j^L\boldsymbol{x}(k)$.

The learning process can be optimized by performance by using a modified Kaczmarz-Widrow-Hoff algorithm [17, 18] in the form of

$$\boldsymbol{w}_j(k) = \begin{cases} \boldsymbol{w}_j(k-1) + \dfrac{e_j(k)\boldsymbol{x}(k)}{\gamma + \|\boldsymbol{x}(k)\|^2}, & -1 \le \mathrm{u}_j(k) \le 1 \\ \boldsymbol{w}_j(k-1) + \dfrac{e_j(k)\boldsymbol{x}^R(k)}{\gamma + \|\boldsymbol{x}^R(k)\|^2}, & \mathrm{u}_j(k) > 1 \\ \boldsymbol{w}_j(k-1) + \dfrac{e_j(k)\boldsymbol{x}^L(k)}{\gamma + \|\boldsymbol{x}^L(k)\|^2}, & \mathrm{u}_j(k) < -1 \end{cases} \quad (13)$$

where $\gamma \ge 0$ is regulatory parameter that protects learning process from zeroing gradient effect.

It is possible to improve approximate properties of neuron with APWLSAF by adjusting not only synaptic weights w_{ij}, but also parameters of activation function a_j^R and a_j^L, while receiving the next vector $\boldsymbol{x}(k)$ on neuron inputs, setup $a_j^R(k)$ and $a_j^L(k)$ is first implemented and then the vector of synaptic weights $\boldsymbol{w}_j(k)$ is refined. That is, the procedure of error backpropagation at individual neurons level is actually implemented. Note, that

$$a_j^R(k) = a_j^R(k-1) + \eta_a(k)e_j(k)\left(u_j(k) - 1\right) = \\ = a_j^R(k-1) + \eta_a(k)\left(y_j(k) - \psi_j\left(w_j^T(k-1)x(k)\right)\right)\left(w_j^T(k-1)x(k) - 1\right), \quad (14)$$

if $u_j(k) > 1$, and

$$a_j^L(k) = a_j^L(k-1) + \eta_a(k)e_j(k)\left(u_j(k) + 1\right) = \\ = a_j^L(k-1) + \eta_a(k)\left(y_j(k) - \psi_j\left(w_j^T(k-1)x(k)\right)\right)\left(w_j^T(k-1)x(k) + 1\right), \quad (15)$$

if $u_j(k) < -1$, then

$$\boldsymbol{w}_j(k) = \begin{cases} \boldsymbol{w}_j(k-1) + \frac{e_j(k)\boldsymbol{x}(k)}{\gamma + \|\boldsymbol{x}(k)\|^2}, & -1 \le \mathrm{u}_j(k) \le 1 \\ \boldsymbol{w}_j(k-1) + \frac{e_j(k)\tilde{\boldsymbol{x}}^R(k)}{\gamma + \|\tilde{\boldsymbol{x}}^R(k)\|^2}, \tilde{\boldsymbol{x}}^R(k) = a_j^R(k)x(k), & \mathrm{u}_j(k) > 1. \\ \boldsymbol{w}_j(k-1) + \frac{e_j(k)\tilde{\boldsymbol{x}}^L(k)}{\gamma + \|\tilde{\boldsymbol{x}}^L(k)\|^2}, \tilde{\boldsymbol{x}}^L(k) = a_j^L(k)x(k), & u_j(k) < -1 \end{cases} \quad (16)$$

4. The experiment results

increasing classification accuracy is very important in problems of image recognition and processing [19 – 22]. An experimental study of the effectiveness of the APWLSAF was carried out to solve the problem of recognizing people's emotions from photographs. The task remains relevant for many applications, where result depends on quality of user interaction and taking into account his emotional status can influence the process [23].

In the experiment some photos from the Extended Cohn-Kanade (CK+) dataset [24] were used. Picked dataset consisting of 821 images of seven emotion classes: anger (126 photos), disgust (67 photos), fear (111 photos), happy (154 photos), sadness (107 photos), surprise (98 photos) and neutral (158 photos). Images were scaled and transformed to grayscale. All experiments were carried out using the TensorFlow framework. Examples of images are shown in Figure 2.

Figure 2: Examples of processed images from Extended Cohn-Kanade (CK+) dataset

Configurable network parameters are changed using standard function ReLU, leakyReLU, PReLU and APWLSAF. Network architecture and training parameters:

- convolution layer contents 20 filters;
- filter size 5×5;
- training cycle consists of 8 and 12 epochs;
- number of iterations 104 and 156;
- iterations per epochs 13;
- minibatch size 32;
- starting value of learning rate 0.0001;
- validation frequency 30 iterations.

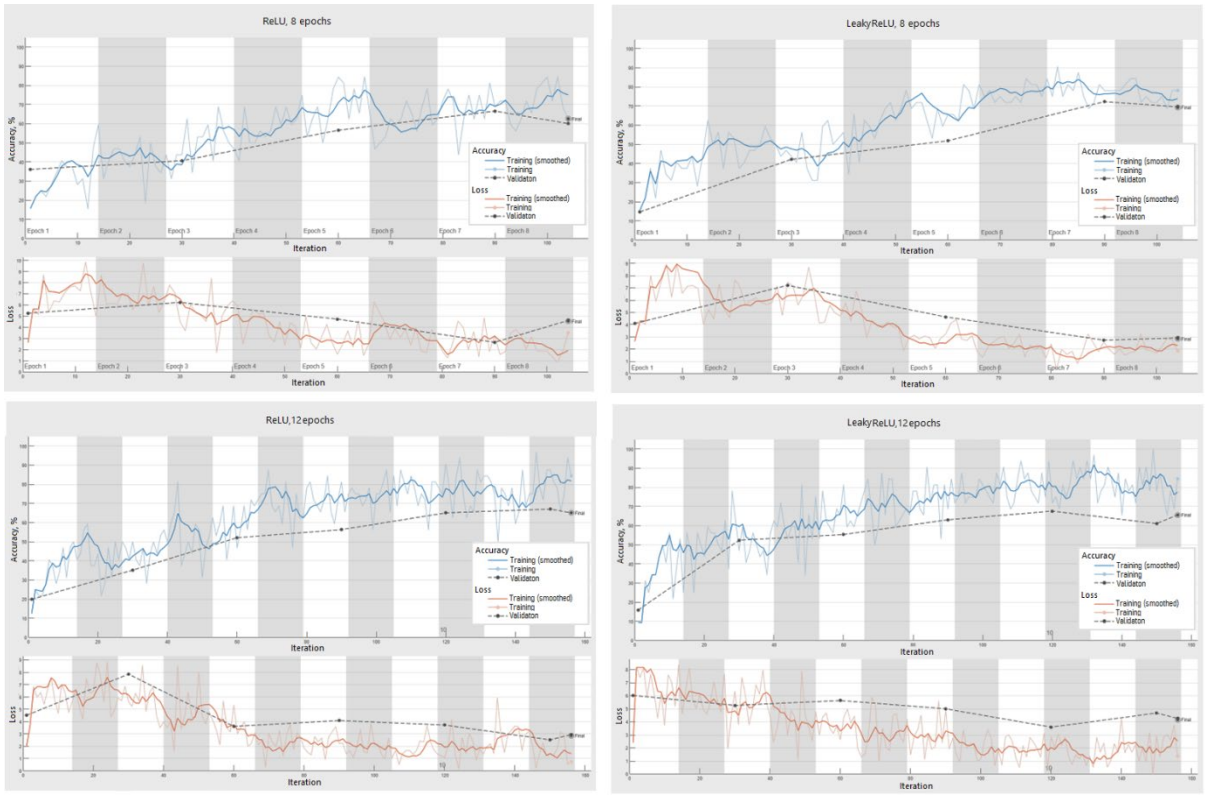
The results of a deep network with different activation functions training are shown in Figures 3-4.

Accuracy of network training with considered activation functions when classifying the emotions in dataset images is shown in the Table 1.

Table 1

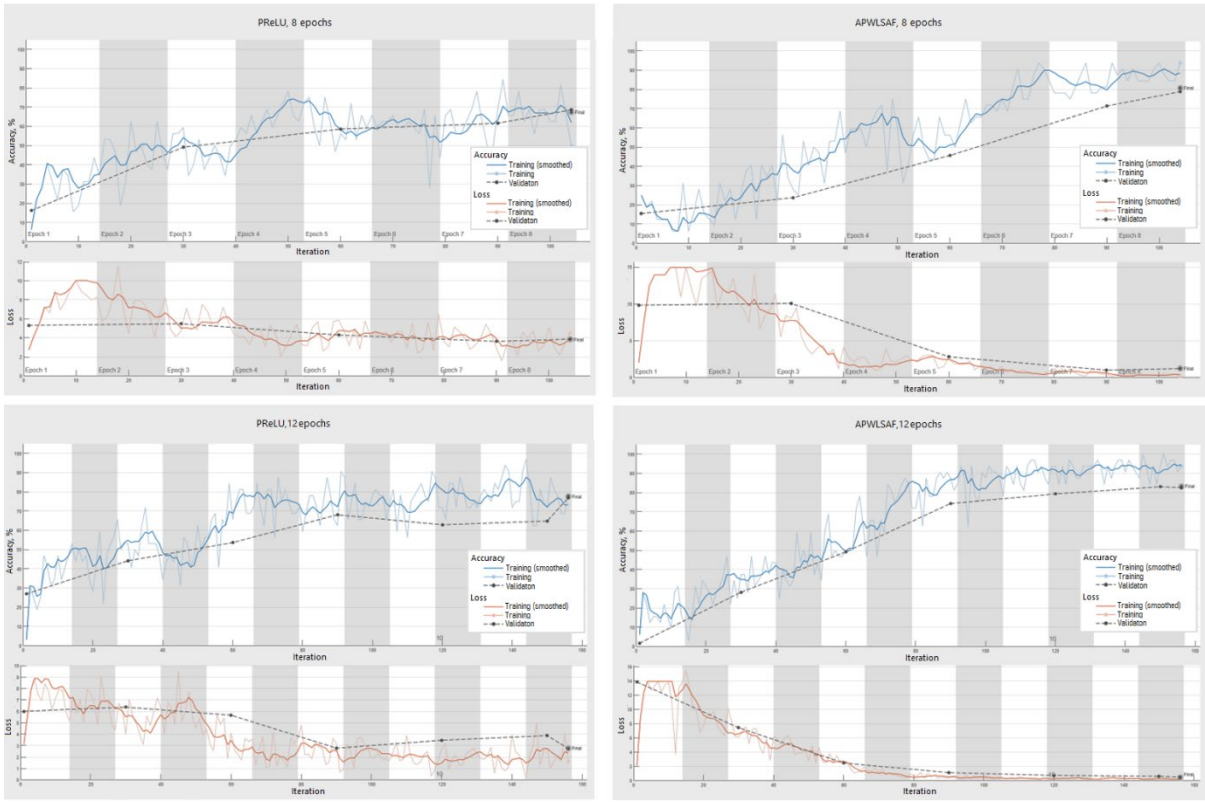
Accuracy of network training with ReLU, leakyReLU, PReLU and APWLS activation functions in emotion classification by images

Activation function	Accuracy (8 learning epochs)	Accuracy (12 learning epochs)
ReLU	62.59%	65.09%
leakyReLU(0.6)	69.32%	65.59%
PreLU	67.33%	77.56%
APWLSAF	80.80%	83.29%



a) b)

Figure 3: Results of network training with a) ReLU activation function; b) leakyReLU activation function



a) b)

Figure 4: Results of network training with a) PReLU activation function; b) APWLSAF

It can be seen that widespread and frequently used activation functions (ReLU, PreLU, leakyReLU) quickly achieve the highest possible accuracy for the task (in the range of 60-70%). Increasing the training duration leads to only a slight increase in accuracy. At the same time, the APWLSAF shows a trend towards increasing accuracy and higher performance (80-85%). Here it is also important to pay attention to the fact that the training dataset was small, and against this background, the gain in accuracy of approximately 15%, which is given by the APWLSAF with all other network parameters unchanged, is an important result. It confirms that the proposed function allows deep networks to learn complex tasks on small data samples faster and more accurately.

5. Conclusion

The new adaptive piece-wise linear squashing activation function (APWLSAF) is proposed, which combines the properties of squashing functions of shallow neural networks (especially three-layer perceptrons) and piece-wise linear functions in deep neural networks without suffering from effect a "vanishing" gradient. The neuron tuning algorithm with APWLSAF, characterized by high speed and ease of numerical implementation, has been introduced. The results of the computer experiment confirm the effectiveness of proposed approach.

References

[1] S.R. Dubey, S.K. Singh, B.B. Chaudhuri, Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark, Neurocomputing 503(11) (2022). doi: 10.1016/j.neucom.2022.06.111.

[2] A. Molina, P. Schramowski, K. Kersting, Padé activation units: end-to-end learning of flexible activation functions in deep networks, arXiv:1907.06732v3 [cs.LG], (2020).

[3] M. Basirat, P.M. Roth, L*ReLU: Piece-wise Linear Activation Functions for Deep Fine-grained Visual Categorization, in 2020 IEEE Winter Conference on Applications of Computer Vision (WACV), Snowmass, CO, USA, March 2020. doi: 10.1109/WACV45572.2020.9093485.

[4] Z. Zhu, Y. Zhou, Y. Dong, Z. Zhong, PWLU: Learning Specialized Activation Functions with the Piecewise Linear Unit, in: IEEE Transactions on Pattern Analysis and Machine Intelligence (Early Access), pp. 1-19. doi: 10.1109/TPAMI.2023.3286109.

[5] S. Dai, S. Mahloujifar, P. Mittal, Parameterizing Activation Functions for Adversarial Robustness, in: 2022 IEEE Security and Privacy Workshops (SPW), 2022. doi: 10.1109/SPW54247.2022.9833884.

[6] M. Loni, A. Mohan, M. Asadi, M. Lindauer, Learning Activation Functions for Sparse Neural Networks, arXiv:2305.10964v2 [cs.LG], (2023).

[7] Y. Ying, J. Su, P. Shan, L. Miao, X. Wang, S. Peng, “Rectified exponential units for convolutional neural networks.” IEEE Access 7 (2019): 101633–101640.

[8] F. Agostinelli, M. Hoffman, P. Sadowski, P. Baldi, Learning activation functions to improve deep neural networks, arXiv:1412.6830v3, available at: www.doi.org/10.48550/arXiv.1412.6830 (2015).

[9] S. Qian, H. Liu, C. Liu, S. Wu, H. San Wong, “Adaptive activation functions in convolutional neural networks.” Neurocomputing 272 (2018): 204–212.

[10] X. Jin, C. Xu, J. Feng, Y. Wei, J. Xiong, S. Yan, Deep learning with s-shaped rectified linear activation units, in: AAAI Conference on Artificial Intelligence, 2016.

[11] S. Gu, W. Li, L. V. Gool, R. Timofte, Fast image restoration with multibin trainable linear units, in: IEEE International Conference on Computer Vision, 2019, pp. 4190–4199.

[12] P. Ramachandran, B. Zoph, Q. V. Le, Searching for activation functions, in: International Conference on Learning Representations Workshops (2018).

[13] S. Eger, P. Youssef, I. Gurevych, Is it time to swish? Comparing deep learning activation functions across nlp tasks, arXiv preprint arXiv:1901.02671 (2019).

[14] Ye. Bodyanskiy, A. Deineko, V. Skorik, F. Brodetskyi. “Deep Neural Network with Adaptive Parametric Rectified Linear Units and its Fast Learning.” International Journal of Computing 21(1) (2022): 11-18. https://doi.org/10.47839/ijc.21.1.2512
[15] K. Hornik, M. Stinchcombe, H. White, “Multilayer feedforward networks are universal approximators.” Neural Networks 2.5 (1989): 359 – 366.
[16] G. Cybenko, “Approximation by superposition of a sigmoidal function.” Math. Control Signals Systems 2 (1989): 303 – 314.
[17] S. Kaczmarz, ”Approximate solution of systems of linear equations”, International Journal of Control, 57:6, 1993, pp. 1269-1271, doi:10.1080/00207179308934446.
[18] B. Widrow, M. Hoff, ”Adaptive Switching Circuits”, IRE WESCON Convention Record, Part 4, pp. 96-104. New York IRE.
[19] S. Khlamov, V. Savanevych, Big astronomical datasets and discovery of new celestial bodies in the Solar System in automated mode by the CoLiTec software, in: Knowledge Discovery in Big Data from Astronomy and Earth Observation (1st ed., part IV), chapt. 18 Astrogeoinformatics, Elsevier, 2020, pp. 331-345. doi: 10.1016/B978-0-12-819154-5.00030-8.
[20] V. Savanevych, S. Khlamov, O. Briukhovetskyi, T. Trunova, I. Tabakova, Mathematical methods for an accurate navigation of the robotic telescopes, in: Mathematics, special issue Mathematics in Robot Control for Theoretical and Applied Problems, vol. 11 issue 10 (2246), 2023. 19 p. doi: 10.3390/math11102246.
[21] V. E. Savanevych, S. V. Khlamov, V. S. Akhmetov, A. B. Briukhovetskyi, V. P. Vlasenko, E. N. Dikov, I. Kudzej, P. A. Dubovsky, D. E. Mkrtichian, I. S. Tabakova, T. O. Trunova. “CoLiTecVS software for the automated reduction of photometric observations in CCD-frames.” Astronomy and Computing 40 (100605) (2022): 15. doi: 10.1016/j.ascom.2022.100605.
[22] V. Akhmetov, S. Khlamov, V. Khramtsov, A. Dmytrenko. “Astrometric reduction of the wide-field images.: Advances in Intelligent Systems and Computing IV: Springer Nature Switzerland 1080 (2020): 896–909. doi: 10.1007/978-3-030-33695-0_58.
[23] N. Kulishova, Emotion Recognition Using Sigma-Pi Neural Network, in: Proceedings of 2016 IEEE First International Conference on Data Stream Mining & Processing (DSMP), Lviv Ukraine, 2016, pp. 327-331.
[24] P. Lucey, J.F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, I. Matthews, "The Extended Cohn-Kanade Dataset (CK+): A complete dataset for action unit and emotion-specified expression", in: Proceedings of IEEE workshop on CVPR for Human Communicative Behavior Analysis, San Francisco, USA, 2010.