НЕЙРОІНФОРМАТИКА ТА ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ

NEUROINFORMATICS AND INTELLIGENT SYSTEMS

UDC 004.8:004.032.26

TUNABLE SQUASHING ACTIVATION FUNCTION FOR DEEP NEURAL NETWORKS

Shafronenko A. Yu. – Dr. Sc., Associate Professor at the Department of Informatics, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine. ROR: https://ror.org/01ctj1b90. ORCID: orcid.org/0000-0002-8040-0279.

Bodyanskiy Ye. V. – Dr. Sc., Professor at the Department of Artificial Intelligence, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine. ROR: https://ror.org/01ctj1b90. ORCID: orcid.org/ 0000-0001-5418-2143.

Shafronenko Ye. O. – Senior Lecturer at the Department of Media Engineering and Information Radio Electronic Systems, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine. ROR: https://ror.org/01ctj1b90. ORCID: orcid.org/0009-0008-0872-2274.

Brodetskyi F. A. – Senior Lecturer at the Department of Informatics, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine. ROR: https://ror.org/01ctj1b90. ORCID: https://orcid.org/0000-0002-0300-3886.

Tanianskyi O. S. – Post-graduate student at the Department of Informatics, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine. ROR: https://ror.org/01ctj1b90. ORCID: orcid.org/0009-0005-3491-4470.

ABSTRACT

Context. At present, artificial neural networks have become widely used to solve many problems of information processing of the most diverse nature and, above all, data mining, due to their universal approximating capabilities and the ability to learn their parameters – synaptic weights. The process of training a multilayer network consists of adjusting the synaptic weights of each neuron using the error backpropagation procedure, which is based on the chain rule of differentiation for complex functions and gradient-based optimization. Deep neural networks are based on multilayer perceptrons, which have proven their effectiveness in solving many very complex problems related to the processing and synthesis of images of various natures, natural language texts, multidimensional stochastic and chaotic sequences, including audio and video signals. Unlike classical three-layer perceptrons, DNNs contain dozens and hundreds of layers, and the number of their synaptic weights is commensurate with or even exceeds the number of synapses in the biological brain. It is clear that for these neural networks, the effect of a vanishing gradient is extremely undesirable; therefore, instead of traditional compression functions, piecewise-linear constructions are usually used here, the most popular of which is the so-called ReLU.

Objective. The purpose of the work is to introduce an adaptive activation function for deep neural networks based on the most common piecewise linear function, ReLU.

Method. A new tunable activation function for deep neural networks is proposed based on the most common piecewise linear function, ReLU, which, however, does not satisfy the conditions of G. Cybenko's approximation theorem, but provides protection for the learning process against the undesirable effect of vanishing gradients.

Results. A new adaptive piecewise linear function based on ReLU is introduced, which is both compressive and protected against vanishing gradients. In this case, during the training process, not only are the synaptic weights adjusted in the network, but also the parameters of the activation function itself. Using the proposed function allows you to reduce the number of neurons and hidden layers in the neural network, the number of required training samples, and the time required to set up the network.

Conclusions. An adaptive squashing activation function based on the widely used ReLU for deep neural networks is introduced, providing both universal approximating properties and preventing vanishing gradients. A training procedure using this function is proposed, offering high performance and a simple numerical implementation. An additional circuit for tuning the parameters of the activation functions can be quite simply introduced into existing deep neural networks that use piecewise linear activation functions.

KEYWORDS: squashing activation function, deep neural network, ReLU, gradient procedures, training signal.

ABBREVIATIONS

ANN – Artificial Neural Networks;
DNN – Deep Neural Networks;
ReLU – Rectified Linear Unit.

NOMENCLATURE

$\hat{y}_j(k)$ – output signal of the j-th neuron of the neural network at the moment of discrete time;
k – discrete time;
N – training sample size;

DOI 10.15588/1607-3274-2026-1-5

$\psi_j(\bullet)$ – nonlinear activation function of the j-th neuron;

$u_j(k)$ – internal activation signal;

$\gamma_j > 0$ – parameter defining the shape of the activation function;

θ_j – bias signal (threshold);

n – number of input signals (dimension of input vectors);

w_{ji} – tunable synaptic weight at the i-th input of the j-th neuron;

$x_i(k)$ – signal at the i-th input of the neuron at time moment k;

δ – learning rule;

$d_j(k)$ – external training signal;

β – smoothing parameter;

E – objective function;

$\eta(k)$ – non-negative learning step parameter.

INTRODUCTION

At present, artificial neural networks are widely used to solve many problems in information processing of the most diverse nature, and above all in data mining, due to their universal approximating capabilities and the ability to learn their parameters – synaptic weights. Multilayer perceptrons have received the widest distribution here. The universal approximating properties of multilayer perceptrons were proven by the theorems of G. Cybenko and K. Hornik [1, 2], using elementary F. Rosenblatt perceptrons as nodes of these neural networks with so-called squashing activation functions, among which the most common are the so-called sigmoidal functions (σ-functions) of the form [1]:

$$\hat{y}_j(k) = \psi_j(u_j(k)) = \frac{1}{1+e^{-\gamma_j u_j(k)}}, \qquad (1)$$

$\gamma_j > 0$ – is a parameter that sets the shape of the activation function and is usually chosen from purely empirical markings, although in principle it can be tuned using a gradient optimization procedure [3].

In this case, the non-linear transformation implemented by a separate neuron node can be written as

$$\hat{y}_j(k) = \psi_j\left(\theta_{jo} + \sum_{i=1}^{m} w_{ji}x_i(k)\right) = \psi_j\left(\sum_{i=1}^{m} w_{ji}x_i(k)\right) =$$
$$= \psi_j\left(w_j^T x(k)\right) = \psi_j\left(u_j(k)\right).$$

REVIEW OF THE LITERATURE

The process of training a multilayer network consists of adjusting the synaptic weights of each neuron using the error backpropagation procedure, which is based on the chain rule of differentiation of complex functions and the gradient optimization procedure (δ – learning rule).

It is easy to see that the derivative of the σ– function (1) has the form

$$\psi_j'(u_j(k)) = \gamma_j \hat{y}_j(k)\left(1-\hat{y}_j(k)\right),$$

and its value decreases as the sigmoid approaches its asymptotes –1 or +1, which is associated with the undesirable effect of the vanishing gradient, which, in turn, leads to the termination of the learning process.

Based on multilayer perceptrons, deep neural networks (DNNs) were developed [4–9], which demonstrated their effectiveness in solving many complex problems related to the processing and synthesis of images of various natures, natural language texts, multidimensional stochastic and chaotic sequences, and audio and video signals. Unlike classical three-layer perceptrons, DNNs contain dozens and hundreds of layers, and the number of their synaptic weights is commensurate with or even exceeds the number of synapses in the biological brain. It is clear that for these neural networks, the effect of a vanishing gradient is extremely undesirable, therefore, instead of traditional squashing functions, piecewise-linear constructions are usually used here, the most popular of which is the so-called ReLU (Rectified Linear Unit), which has the form:

$$\hat{y}_j(k) = \psi_j(u_j(k)) = \begin{cases} u_j, & u_j \geq 0, \\ 0 \text{ otherwise.} \end{cases} \qquad (2)$$

It is clear that there is no vanishing gradient here, and the extremely simple derivative of (2) allows accelerating the learning process of a separate neuron, since

$$\psi_j'(u_j(k)) = \begin{cases} 1, & u_j \geq 0, \\ 0 \text{ otherwise.} \end{cases}$$

At the same time, function (2) is not squashing, i.e., it does not satisfy the conditions of Cybenko's theorem, which means that to ensure the required quality of approximation, the number of neurons with such functions must be very large (recall integration by the trapezoidal method). Obviously, the gain in time when training a separate neuron is completely leveled by the huge number of these neurons in the network.

Therefore, it is advisable to introduce a piecewise-linear function (simple derivatives) that is simultaneously squashing (approximating properties) and at the same time protected from the vanishing gradient.

MATERIALS AND METHODS

Based on ReLU, the graph of which is shown in Figure 1.

DOI 10.15588/1607-3274-2026-1-5

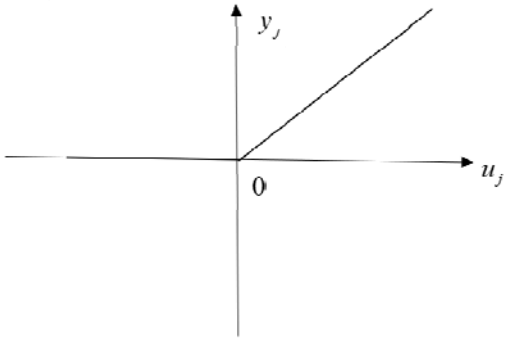

Figure 1 – ReLU Activation Function

It is not difficult to introduce a piecewise-linear function of the form:

$$\hat{y}_j(k) = \psi_j(u_j(k)) = \begin{cases} u_j, & 0 \le u_j \le 1, \\ 1, & u_j > 1, \\ 0, & u_j < 0, \end{cases}$$

the graph of which is shown in Figure 2.

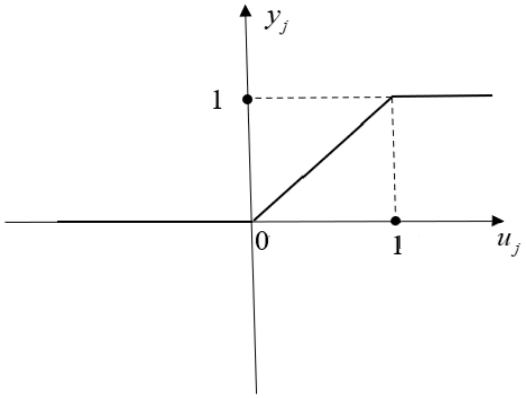

Figure 2 – Piecewise-linear squashing activation function

The use of this function in neural networks trained via gradient-based optimization procedures is impractical, as it immediately encounters the vanishing gradient problem.

Therefore, it is proposed to introduce a function of the form:

$$\hat{y}_j(k) = \psi_j(u_j(k)) = \begin{cases} u_j, & 0 \le u_j \le 1, \\ 1 - a\left(1 - u_j\right), & u_j > 1, \\ 0, & u_j < 0, \end{cases}$$

the graph of which is shown in Figure 3.

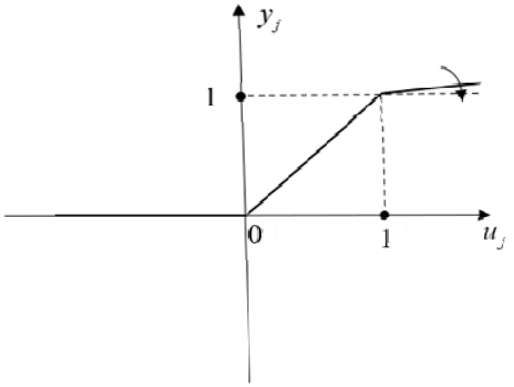

Figure 3 – Piecewise-linear activation function protected from vanishing gradient

The derivative of this function at $u_j(k) > 1$ has the form

$$\psi'_j(u_j(k)) = a,$$

at the same time, by imposing a threshold $a \ge \varepsilon$, it is possible to avoid stopping the learning process. Moreover, by simultaneously excluding the parameter and the synaptic weights, it is possible to arrive at this process.

As a target learning function, we will use the local quadratic criterion:

$$E_j(k) = \frac{1}{2} e_j^2(k) = \frac{1}{2}\left(d_j(k) - \hat{y}_j(k)\right)^2. \qquad (3)$$

The process of gradient optimization of criterion (3) by synaptic weights is implemented using the standard δ-rule [11], which in this case can be written as:

$$\begin{aligned} w_{ji}(k+1) &= w_{ji}(k) - \eta(k)\frac{\partial E_j(k)}{\partial w_{ji}} = \\ &= w_{ji}(k) - \eta(k)\frac{\partial E_j(k)}{\partial e_j(k)} \bullet \frac{\partial e_j(k)}{\partial w_{ji}} = \\ &= w_{ji}(k) - \eta(k) e_j(k)\frac{\partial e_j(k)}{\partial w_{ji}} = \\ &= w_{ji}(k) - \eta(k) e_j(k)\frac{\partial e_j(k)}{\partial u_{ji}} \bullet \frac{\partial u_j(k)}{\partial w_{ji}} = \\ &= w_{ji}(k) + \eta(k) e_j(k)\psi'\left(u_j(k)\right) x_i(k) = \\ &= w_{ji}(k) + \eta(k)\delta_j(k) x_i(k). \end{aligned} \qquad (4)$$

Procedure (4) can also be written in a simple form

$$w_{ji}(k+1) = w_{ji}(k) + \eta(k)\delta_j(k)x(k),$$

and since

DOI 10.15588/1607-3274-2026-1-5

$$\psi'_j(u_j(k)) = \begin{cases} 1, & 0 \le u_j \le 1, \\ a, & u_j > 1, \end{cases}$$

the learning process can be optimized for speed using the Kaczmarz-Withrow-Hoff algorithm [12–14], which in this case takes the form

$$w_{ji}(k+1) = w_{ji}(k) + \frac{\partial e_j(k)\psi'_j\left(u_j(k)\right)x(k)}{\|x(k)\|^2} = \\ = w_j(k) + \frac{\delta_j(k)x(k)}{\|x(k)\|^2}. \quad (5)$$

The procedure with additional smoothing properties [15, 16] in a modified form can also be used:

$$\begin{cases} w_j(k+1) = w_j(k) + r^{-1}(k)\delta_j(k)x(k), \\ r(k+1) = \beta r(k) + \|x(k)\|^2, \end{cases} \quad (6)$$

where $0 \le \beta \le 1$.

The quality of the learning process can be improved by additionally tuning the parameter a at $u_j(k) > 1$, while controlling the fulfillment of the inequality $a \ge \varepsilon$. At the same time, at each tuning cycle of the neuron, the value is first adjusted $a_j(k)$ at $u_j(k) > 1$, and then the synaptic weights are adjusted:

$$\begin{cases} a_j(k+1) = a_j(k) + \eta(k)e_j(k)\left(u_j(k) - 1\right), \\ w_j(k+1) = w_j(k) + \eta(k)e_j(k)\psi'_j\left(u_j(k)\right)x(k), \end{cases}$$

where

$$\psi'_j(u_j(k)) = \begin{cases} 1, & 0 \le u_j \le 1, \\ a_j(k+1), & u_j(k) > 1, \\ 0, & u_j(k) < 0. \end{cases}$$

Thus, a local error backpropagation procedure is implemented at the level of a single neuron. Training of the neural network as a whole is implemented using the error backpropagation rule, with an additional loop for tuning the parameters of the activation functions.

EXPERIMENTS

During the experimental research, it was decided to classify images from the "Fashion-MNIST" dataset.

Examples of images are shown in Figure 4.

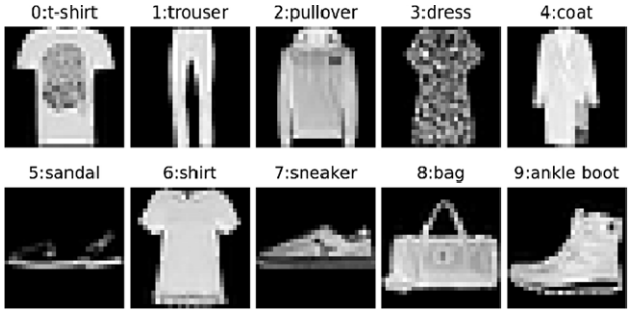

Figure 4 – Exemplars of the Fashion MNIST sample

For experimental research, the sample was previously divided into two subgroups, with the type and depth of the neural network also taken into account. Thus, in the first subgroup, the performance of activation functions is evaluated with a fixed neural network architecture and varying depth. The second subgroup evaluates activation functions based on different types of the most popular DNNs.

Note also that CNN is used across 3 depth variants: CNN1, CNN2, and CNN3. Comparative analysis was carried out on 4 activation functions. The CNN1 architecture consists of two convolution layers, one fully connected layer, and one softmax classification layer. Similarly, the structure of CNN2 is arranged the same as CNN1, but with two additional convolutional layers. CNN3 is deeper, with six convolutional layers and two additional convolutional layers. All experiments account for the average performance across five data runs.

Comparative accuracy results are shown in Table 1. The last column of Table 1 displays the average accuracy for each activation function across all three CNN models.

Table 1 – Accuracy testing results on Fashion-MNIST in %

Activation Function	CNN 1	CNN 2	CNN 3	Average Accuracy
ReLU	**93.5**	93.9	94.2	93.9
LeakyReLU	93	92.9	**93.7**	93.2
Piecewise linear ReLU	**93.5**	**94.2**	94.3	**94**

Let's conduct an experiment with activation functions on various models of popular DNNs: ResNet56V1 (1), ResNet56V2 (2), and ResNet110 (3). First, image benchmarks are tested at different network depths. Secondly, benchmark images are checked on other models of common architectures. The results of the experiments are shown in Table 2.

Table 2 – Accuracy testing results on Fashion-MNIST in %

Activation Function	1	2	3	Average Accuracy
ReLU	88.9	90.1	90.4	89.8
LeakyReLU	89	90.6	90.3	90
Piecewise linear ReLU	**90**	**91.2**	**90.8**	**90.6**

DISCUSSION

The first stage of the analysis aims to evaluate the overall performance of all activation functions across image and text classification tasks. The second stage aims to determine the most successful activation functions across all launched models. Experimental studies show that the proposed adaptive piecewise-linear function achieves the

DOI 10.15588/1607-3274-2026-1-5

best accuracy in CNN1 and CNN2 and is independent of the DNN model type, as demonstrated in Table 2.

In the first stage of analysis, we average each activation function across all launched models using general image benchmarks. Figure 5 summarizes the average accuracy for all image benchmarks.

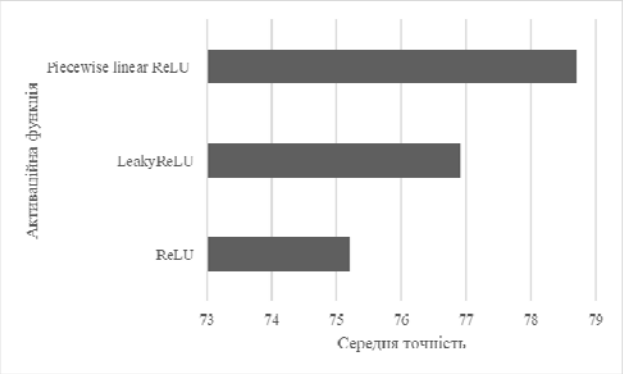

Figure 5 – Average accuracy value of activation functions compared to others, using different popular networks based on the "Fashion MNIST" sample

In the second stage of analysis, an idea of the most successful activation function for each model was obtained. Table 3 shows the most successful combinations between activation functions and launched models.

Table 3 – Successful activation functions for each deep network

Activation Function	CNN1	CNN2	CNN3	1	2	3
ReLU	+					
LeakyReLU			+			
Piecewise linear ReLU	+	+		+	+	+

CONCLUSIONS

An adaptive squashing activation function based on the widely used ReLU for deep neural networks is introduced, which simultaneously provides universal approximating properties while preventing the undesirable vanishing gradient problem. A training procedure using this function is proposed, ensuring high speed and a simple numerical implementation. An additional circuit for tuning the parameters of activation functions can be quite simply introduced into existing deep neural networks that use piecewise-linear activation functions.

Scientific novelty: A training procedure using an adaptive activation function for deep neural networks is proposed, ensuring high speed and a simple numerical implementation.

Practical significance: Using the proposed function reduces the number of neurons and hidden layers in the neural network, the required volume of training samples, and the time required to set up the network.

Prospects for further research: Fast neural networks for pattern/image recognition for a wide class of practical tasks in Data Stream Mining and Big Data Mining.

ACKNOWLEDGEMENTS

The work is supported by the state budget scientific research project of Kharkiv National University of Radio Electronics "Adaptive bagging of hybrid computational intelligence systems based on speed-optimal online learning" (SR No. 0124U000363).

DECLARATIONS

Conflict of interest: The authors declare that they have no conflict of interest in relation to this research, whether financial, personal, authorship, or otherwise, that could affect the research and its results presented in this paper.

Author's contributions: A. Shafronenko to introduce an adaptive activation function for deep neural networks based on the most common piecewise linear function, ReLU; Ye. Bodyanskiy proposed a gradient optimization process for the quadratic criterion over synaptic weights; F. Brodetskyi adjusted the parameter that determines the shape of the activation function; Ye. Shafronenko evaluated the overall performance of activation functions across image and text classification tasks; O. Tanianskyi conducted experimental studies using activation functions with various models of popular GNMs.

Data availability: The manuscript has associated data in a data repository https://openarchive.nure.ua/.

Software availability: The manuscript has no associated software.

Use of artificial intelligence tools: The authors confirm that they did not use artificial intelligence technologies in creating the submitted work.

REFERENCES

1. Cybenko G. Approximation by superpositions of a sigmoidal function, *Mathematics of control, signals and systems*, 1989, 2.4, pp. 303–314.
2. Hornik K., Stinchcombe M., White H. Multilayer feedforward networks are universal approximators, *Neural Networks*, 1989, 2.5, pp. 359–366.
3. Hornik K. Approximation capabilities of multilayer feedforward networks, *Neural Networks*, 1991, 4.2, pp. 251–257.
4. Poggio T., Girosi F. Networks for approximation and learning, *Proceedings of the IEEE*, 1990, Vol. 78, № 9, pp. 1481–1497.
5. Haykin S. Neural networks: a comprehensive foundation. Prentice Hall PTR, 2004. Vol. 2. 1994.
6. Vapnik V. N. The Nature of Statistical Learning Theory. New York, Springer, 1995.
7. Cortes C. and Vapnik V. Support-vector networks, *Machine Learning*, Sep. 1995, Vol. 20, No. 3, pp. 273–297, https://doi.org/10.1007/bf00994018.
8. Bodyanskiy Ye., Zaychenko Yu. and Hamidov G. Hybrid Deep Learning Networks Based on Self-Organization and their Applications. Cambridge Scholars Publishing, 2024.
9. Kaczmarz S. Approximate solution of systems of linear equations, *International Journal of Control,* 1993, No. 57 (6), pp. 1269–1271.
10. Widrow B. and Hoff M. E. Adaptive switching circuits, *1960 IRE WESCON Convention Record*, 1960, pp. 96–104.
11. Bodyanskiy Ye., Kolodyazhniy V. and Stephan A. An adaptive learning algorithm for a neuro-fuzzy network, *International Conference on Computational Intelligence*. Berlin, Heidelberg, Springer Berlin Heidelberg, 2001, pp. 68–75.

Received 08.10.2025.
Accepted 30.01.2025.
Published 27.03.2026.

DOI 10.15588/1607-3274-2026-1-5

УДК 004.8:004.032.26

НАЛАШТОВНА СТИСКАЮЧА АКТИВАЦІЙНА ФУНКЦІЯ ДЛЯ ГЛИБОКИХ НЕЙРОННИХ МЕРЕЖ

Шафроненко А. Ю. – д-р техн. наук, доцент кафедри інформатики Харківського національного університету радіоелектроніки, Харків, Україна. ROR: https://ror.org/01ctj1b90. ORCID: orcid.org/0000-0002-8040-0279.

Бодянський Є. В. – д-р техн. наук, проф., проф. кафедри штучного інтелекту, Харківський національний університет радіоелектроніки, Харків, Україна. ROR: https://ror.org/01ctj1b90. ORCID: orcid.org/ 0000-0001-5418-2143.

Шафроненко Є. О. – старший викладач кафедри медіаінженерії та інформаційних радіоелектронних систем, Харківський національний університет радіоелектроніки, Харків, Україна. https://ror.org/01ctj1b90. ORCID: orcid.org/0009-0008-0872-2274.

Бродецький Ф. А. – старший викладач кафедри інформатики, Харківський національний університет радіоелектроніки, Харків, Україна. ROR: https://ror.org/01ctj1b90. ORCID: https://orcid.org/0000-0002-0300-3886.

Танянський О. С. – аспірант кафедри інформатики, Харківський національний університет радіоелектроніки, Харків, Україна. ROR: https://ror.org/01ctj1b90. ORCID: orcid.org/0009-0005-3491-4470.

АНОТАЦІЯ

Актуальність. На цей час штучні нейронні мережі отримали широке поширення для вирішення багатьох задач опрацювання інформації найрізноманітнішої природи і, перш за все, інтелектуального аналізу даних, завдяки своїм універсальним апроксимуючим можливостям і здатності до навчання своїх параметрів – синаптичних ваг. Процес навчання багатошарової мережі полягає у налаштуванні синаптичних ваг кожного нейрона за допомогою процедури зворотного поширення похибок, яка базується на ланцюговому правилі диференціювання складних функцій та градієнтної процедури оптимізації. На основі багатошарових персептронів були створені глибокі нейронні мережі, що довели свою ефективність при вирішенні багатьох дуже складних задач, пов'язаних з обробкою і синтезом зображень різноманітної природи, природномовних текстів, багатовимірних стохастичних і хаотичних послідовностей, включаючи аудіо та відеосигнали. На відміну від класичних трьошарових персептронів ГНМ містять десятки та сотні шарів, а кількість їх синаптичних ваг є співрозмірною або навіть перевищує кількість синапсів у біологічному мозку. Зрозуміло, що для цих нейронних мереж ефект зникаючого градієнта є вкрай небажаним, тому замість традиційних стискаючих функцій тут використовуються зазвичай кусково-лінійні конструкції, найбільш популярною з яких є, так звана, ReLU.

Мета. Мета роботи полягає у запровадженні адаптивної активаційної функції для глибоких нейронних мереж на основі найбільш розповсюдженої кусково-лінійної функції ReLU (Piecewise linear ReLU).

Метод. Запропонована нова налаштовна активаційна функція для глибоких нейронних мереж на основі найбільш розповсюдженої кусково-лінійної функції ReLU, яка однак не відповідає умовам апроксимаційної теореми Дж. Цибенка, але забезпечує захист процесу навчання від небажаного ефекту зникаючого градієнта.

Результати. Введено нову адаптивну кусково-лінійну функцію на основі ReLU (Piecewise linear ReLU), що одночасно є як стискаючою, так і захищеною від зникаючого градієнта. При цьому у процесі навчання у мережі налаштовуються не лише синаптичні ваги, але і параметри самої активаційної функції. Використання запропонованої функції дозволяє скоротити кількість нейронів та прихованих шарів у нейронній мережі, необхідний обсяг навчальних вибірок та час налаштування мережі в цілому.

Висновки. Введено у розгляд адаптивну стискаючу активаційну функцію на основі широко поширеної ReLU для глибоких нейронних мереж, що одночасно забезпечує універсальні апроксимуючі властивості і в той же час мережа не потерпає від небажаного ефекту зникаючого градієнта. Запропонована процедура навчання з використанням цієї функції, що забезпечує високу швидкодію та характеризується простотою чисельної реалізації. Додатковий контур налаштування параметрів активаційних функцій досить просто може бути введений у вже існуючі глибокі нейронні мережі, що використовують кусково-лінійні активаційні функції.

КЛЮЧОВІ СЛОВА: адаптивна стискаюча активаційна функція, глибока нейронна мережа, ReLU, градієнтні процедури, навчальний сигнал.

ЛІТЕРАТУРА

1. Cybenko G. Approximation by superpositions of a sigmoidal function / G. Cybenko // Mathematics of control, signals and systems, – 1989. – 2.4. – P. 303–314.
2. Hornik K. Multilayer feedforward networks are universal approximators / K. Hornik, M. Stinchcombe, H. White // Neural networks. – 1989. – 2.5. – P. 359–366.
3. Hornik K. Approximation capabilities of multilayer feedforward networks / K. Hornik // Neural networks. – 1991. – 4.2. – P. 251–257.
4. Poggio T. Networks for approximation and learning / T. Poggio, F. Girosi // Proceedings of the IEEE. – 1990. – T. 78, № 9. – P. 1481–1497.
5. Haykin S. Neural networks: a comprehensive foundation / S. Haykin. – Prentice Hall PTR, 2004. – T.2. –1994.
6. Vapnik V. N. The Nature of Statistical Learning Theory / V. N. Vapnik. – New York : Springer, 1995.
7. Cortes C. Support-vector networks / C. Cortes and V. Vapnik // Machine Learning. – Sep. 1995. – Vol. 20, No. 3. – P. 273–297, https://doi.org/10.1007/bf00994018.
8. Bodyanskiy Ye. Hybrid Deep Learning Networks Based on Self-Organization and their Applications. / Ye. Bodyanskiy, Yu. Zaychenko and G. Hamidov. – Cambridge Scholars Publishing, 2024.
9. Kaczmarz S. Approximate solution of systems of linear equations / S. Kaczmarz // International Journal of Control. – 1993. – No. 57 (6). – P. 1269–1271.
10. Widrow B. Adaptive switching circuits / B. Widrow and M. E. Hoff // In 1960 IRE WESCON Convention Record. – 1960. – P. 96–104.
11. Bodyanskiy Ye. An adaptive learning algorithm for a neuro-fuzzy network / Ye. Bodyanskiy, V. Kolodyazhniy and A. Stephan // International Conference on Computational Intelligence. – Berlin, Heidelberg : Springer Berlin Heidelberg, 2001. – P. 68–75.

DOI 10.15588/1607-3274-2026-1-5