Discrete & Continuous Models
& Applied Computational Science
ISSN 2658-7149 (Online), 2658-4670 (Print)

2025, 33 (3) 242–259
http://journals.rudn.ru/miph

Research article

UDC 004.032.26
PACS 07.05.Mh, 07.05.Kf
DOI: 10.22363/2658-4670-2025-33-3-242-259 EDN: HFFBMV

Construction and modeling of the operation of elements of computing technology on fast neurons

Mikhail V. Khachumov[1, 2], Yuliya G. Emelyanova[3], Vyacheslav M. Khachumov[1, 2, 3]

[1] RUDN University, 6 Miklukho-Maklaya str., Moscow, 117198, Russian Federation

[2] Federal Research Center "Computer Science and Control" of Russian Academy of Sciences, 9 60-let Octyabrya prosp., Moscow, 117312, Russian Federation

[3] Ailamazyan Program Systems Institute of RAS, 4a Peter the Great str., Pereslavl-Zalessky, Yaroslavl Region, 152021, Russian Federation

(received: June 29, 2025; revised: July 20, 2025; accepted: July 25, 2025)

Abstract. The article is devoted to the construction of fast neurons and neural networks for the implementation of two complete logical bases and modeling of computing devices on their basis. The main idea is to form a fast activation function based on semi-parabolas and its variations that have effective computational support. The constructed activation functions meet the basic requirements that allow configuring logical circuits using the backpropagation method. The main result is obtaining complete logical bases that open the way to constructing arbitrary logical functions. Models of such elements as a trigger, a half adder, and an adder, which form the basis of various specific computing devices, are presented and tested. It is shown that the new activation functions allow obtaining fast solutions with a slight decrease in quality compared to reference outputs. To standardize the outputs, it is proposed to combine the constructed circuits with a unit jump activation function.

Key words and phrases: new activation functions, parabola, full logical basis, element models, performance, experimental studies

For citation: Khachumov, M. V., Emelyanova, Y. G., Khachumov, V. M. Construction and modeling of the operation of elements of computing technology on fast neurons. *Discrete and Continuous Models and Applied Computational Science* **33** (3), 242–259. doi: 10.22363/2658-4670-2025-33-3-242-259. edn: HFFBMV (2025).

1. Introduction

The need to increase the speed of artificial neural networks (ANN) containing a huge number of neurons leads to the construction of "fast neurons", which is achieved by a special implementation of activation functions [1–3] and their hardware implementation. This has led to the expansion of research in the field of creating and studying new activation functions and their practical use. Currently, great efforts are aimed at accelerating the operation of activation functions for the construction of "fast" neurons and neural networks based on them. A comparison of various typical activation functions (AFs) in the ANN is performed in the works [4–6]. The issues of searching for methods to reduce the computational complexity of implementing AFs are discussed in [7–9].

Of interest is the creation of elements of computer technology (CT) using complete logical bases (complete systems of logical functions). These are, for example, ∧, ∨, ¬ (conjunction, disjunction, negation); ∧, ¬ (conjunction, negation); ∨, ¬ (disjunction, negation). The devices are implemented by covering them with neurons and neural networks. Interest in this approach is not weakening and is currently growing with the increasing interest in artificial intelligence technologies. It is worth noting the emergence of a class of specialized processors that serve to accelerate the work of neurons and the training algorithms of ANNs as a whole [10].

In this paper, we propose to implement typical elements of the VT on "fast" neurons obtained by replacing known activation functions with new functions that are characterized by a higher implementation speed. Previously, the authors proposed a new activation function called the s-parabola, created a complete logical basis and algorithms for setting up such neurons and neural networks [11, 12].

In this paper, the main focus is on the construction of fast neurons using new activation functions s-parabola, Sparabola-ReLU and ReLU-Sparabola, which are combinations of parabolas and linear functions. Based on them, fast VT elements are proposed: "XOR", trigger, half adder, adder.

Structure of the article

In section 2. Materials and Methods in paragraph 2.1. The basic requirements for activation functions are presented. Section 2.2. Standard and new activation functions propose new functions that can be used in ANNs configured using the backpropagation method. The settings and quality parameters of neurons that form the logical bases "AND"–"OR"–"NOT" and the bases "AND-NOT", "OR-NOT" are shown. The settings are linked to the basic neuron schemes (one neuron, two neurons, constructor) for configuring activation functions.

In section 3. Results in paragraph 3.1. show the settings of the activation functions on the logical basis "AND"–"OR"–"NOT" by the method of back propagation of the error. In relation to the basic circuits of neuron connection, comparative estimates of the speed of tuning and implementation of functions are given. Similar studies are performed in section 3.2. for tuning activation functions to the logical bases "AND-NOT", "OR-NOT". To expand the studies in section 3.3., new activation functions are additionally tuned to the XOR function. Section 3.4. considers the construction of VT elements on logical bases. The results of covering the circuits of the RS trigger of the half adder and adder with logical elements on neurons and the "XOR" function are shown.

In section 4. Discussion discusses the obtained results in comparison with the state of the world level of development of the subject area. The prospects of the proposed approach are associated with the use of fast activation functions in multilayer and convolutional ANNs. It is noted that significant acceleration can be achieved by switching to hardware implementation of activation functions in the form of bit-parallel circuits and the use of CORDIC family algorithms.

In section 5. Conclusion summarizes the results obtained and provides suggestions for further research development.

2. The materials and methods

2.1. Basic requirements for activation functions. Complete logical basis on traditional and fast neurons

The requirements for activation functions are quite contradictory, but at the same time, the following features can be highlighted.

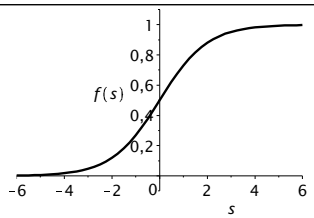
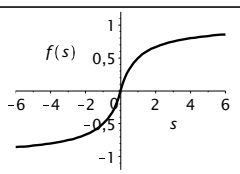
1. If neural networks are trained using the backpropagation method using the gradient descent process, then the layers in the model and the activation functions must be differentiable. Some activation functions, for example, linear or hyperbolic tangent, are differentiable over the entire range of admissible values.
2. The requirements for the activation function of an ANN are determined by a special theorem on completeness [13], according to which the function must be twice differentiable and continuous. The derivative of the activation function must be defined on the entire abscissa axis. To be used in a neuron, the function must be monotonically increasing or decreasing, have parameters that can be adjusted during the training process.
3. It is desirable that the output of the activation function be symmetrical with respect to zero, so that the gradients do not shift in a certain direction. Such a case corresponds, for example, to the sigmoid rational.
4. For logical problems, the excitation values of the output layer neurons must belong to the range [0, 1]. This corresponds, for example, to the sigmoid function and the unit jump (step function). However, the unit jump function does not meet the requirements of differentiability. It is not differentiable at point 0, and its derivative is 0 at all other points. Gradient descent methods do not work for such a function. This can create problems with training, since the numerical gradients calculated near the point where the derivative does not exist may be incorrect.
5. Since activation functions must be calculated repeatedly in deep networks, their calculation must be inexpensive in computational terms. This fact requires a revision of the implementation methods and, possibly, the creation of new functions, which we will call fast-acting.

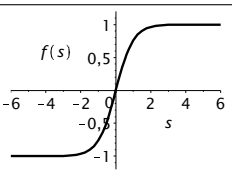
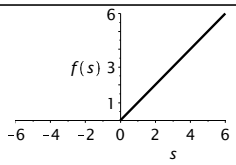
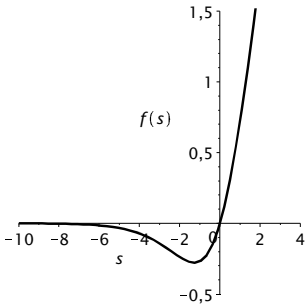
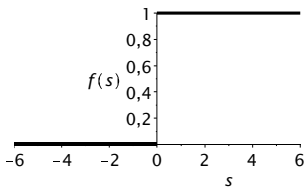
2.2. Typical and new activation functions

Table 1 presents activation functions that have found wide application in neural networks.

Table 1

Typical activation functions

Name	Formula	Schedule
Sigmoid	$f(s) = \frac{1}{1 + e^{-\alpha s}}$	
Sigmoid-rational function	$f(s) = \frac{s}{1 + \lvert s \rvert}$	

Hyperbolic tangent	$f(s) = \frac{e^{2s} - 1}{e^{2s} + 1}$	
ReLU	$f(s) = max(0, s)$	
SiLu	$f(s) = \frac{s}{1 + e^{-s}}$	
Single jump	$f(s) = \begin{cases} 1 & \text{if } s > 0 \\ 0 & \text{if } s \leq 0 \end{cases}$	

The approach proposed in this paper is based on the idea.

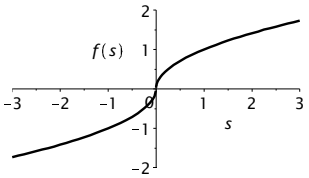
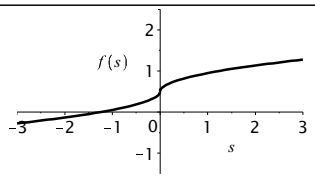
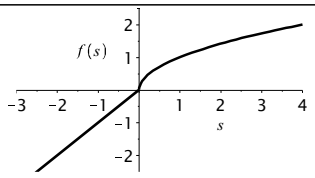
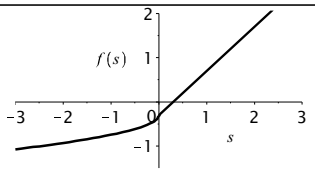
1. Construction of fast neuron models based on the activation function of the "s-parabola" type and their application in individual ANNs and classifier committees. The "s-parabola" function has a structure in which the upper part (the first quarter) is the upper branch of the parabola, and the lower part is a mirror image of the lower part of the parabola relative to the ordinate axis (the third quarter). The graph and formula of the proposed activation function are presented in Table 2 .
2. Combining different types of activation functions to achieve an effective solution to the problem.

The prospects and advantage of the s-parabola are associated with the simplicity of calculating the function, which ensures the speed of implementation. The s-parabola can be used as an activation function of the ANN, since it satisfies the established requirements of twice differentiability. A similar function can be used in multilayer direct propagation ANNs to solve problems of recognizing rigid objects and predicting time processes.

Table 2 presents some new neuronal activation functions.

Table 2

New activation functions: s-parabola and its variations

Name	Formula	Parameters	Schedule
S-parabola	$f(s)=\begin{cases}\beta+\sqrt{2ps} & \text{if } s>0\\ \beta-\sqrt{2ps} & \text{if } s\le 0\end{cases}$	$p=\frac{1}{3}$ $\beta=0.0$	
S-parabola	$f(s)=\begin{cases}\beta+\sqrt{2ps} & \text{if } s>0\\ \beta-\sqrt{-2ps} & \text{if } s\le 0\end{cases}$	$p=0.1$ $\beta=0.5$	
ReLU-Sparabola	$f(s)=\begin{cases}\beta+\sqrt{2ps} & \text{if } s>0\\ s+\beta & \text{if } s\le 0\end{cases}$	$p=0.5$ $\beta=0.01$	
Sparabola-ReLU	$f(s)=\begin{cases}s+\beta & \text{if } s>0\\ \beta-\sqrt{-2ps} & \text{if } s\le 0\end{cases}$	$p=0.1$ $\beta=-0.3$	

Let us consider the construction of a complete logical basis on fast neurons. The implementation of the functions “AND”, “OR”, “NOT”, “AND–NOT”, “OR–NOT” is carried out on the basis of basic circuits with one and two neurons Table 3 .

2.3. Performance evaluation of activation functions

Let n be the bit depth of the numbers being processed, then we can estimate the complexity of executing the activation functions (Table 3).

Table 3

Activation functions sorted in order of decreasing complexity

Function	Computational complexity	Rating (for $n=8$)
SiLu	$n^2((\log n)^2+\log n+1)+n$	840
Sigmoid	$n^2(1+(\log n)^2+\log n)$	832
Hyperbolic tangent	$n^2((\log n)^2+\log n)+2n+1$	785

Sigmoid-rational	$n(1 + n \log n)$	200
S-parabola	$2n^2$	128
ReLU-Sparabola	$n, n + 2n^2$	(8, 136), average 72
Sparabola-ReLU	$n + 2n^2, n$	(136, 8), average 72
ReLU	$n + 1$	9

For ReLU-Sparabola, Sparabola-ReLU the complexity of the implementation depends on which part of the function is executed (left or right).

2.4. Structural diagrams of neurons and their constructs

Variants of neuron schemes are presented in Fig. 1.

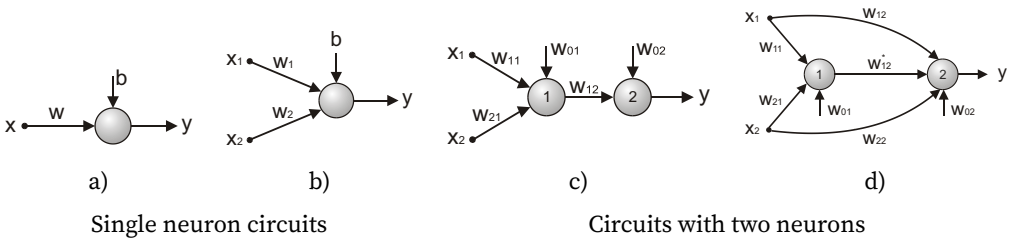


Figure 1. Basic neuron schemes for setting up activation functions

3. Results

The activation functions are configured using the backpropagation method for the logical functions “OR”, “AND”, “NOT” (Table 4-6).

The calculation of speed characteristics was performed on a personal computer with the following parameters: processor: Intel Core i5-6600K @ 3.50 GHz; RAM: 32 GB.

3.1. Activation function settings on the logical basis “AND”–“OR”–“NOT”

Table 4

Comparative characteristics of the setting for the “OR” function on one neuron

Characteristics of training	S-parabola	ReLU-Sparabola	Sparabola-ReLU	Sigmoid
Starting weights	$w_1 = 1, w_2 = 1, b = -0.5$			
Setup results	$w_1 = 1.385$, $w_2 = 1.351$, $b = -0.516$	$w_1 = 4.963$, $w_2 = 5.071$, $b = 3.812$	$w_1 = 0.475$, $w_2 = 0.483$, $b = -0.004$	$w_1 = 15.685$, $w_2 = 15.712$, $b = -6.967$
p	0.1	0.1	2.0	—
β	0.5	-0.5	0.25	—

α	—	—	—	0.5
Average deviation	0.124	0.189	0.181	0.009
Training time, ms	648	7307	27072	1483
Time to implement 10000000 data table cycles, ms	621	645	578	1040

Table 5

Comparative characteristics of the setting for the “AND” function on one neuron

Characteristics of training	S-parabola	ReLU-Sparabola	Sparabola-ReLU	Sigmoid
Starting weights	$w_1 = 1, w_2 = 1, b = -1.5$			
Setup results	$w_1 = 1.268$, $w_2 = 1.359$, $b = -1.756$	$w_1 = 0.516$, $w_2 = 0.405$, $b = 0.098$	$w_1 = 0.874$, $w_2 = 0.874$, $b = -0.514$	$w_1 = 16.338$, $w_2 = 16.338$, $b = -24.899$
p	0.1	1.75	0.1	—
β	0.5	-1.0	-0.3	—
α	—	—	—	0.5
Average deviation	0.123	0.274	0.098	0.010
Training time, ms	5942	602	34332	3159
Time to implement 10000000 data table cycles, ms	621	636	562	1053

Table 6

Comparative characteristics of the setting for the “NOT” function on one neuron

Characteristics of training	S-parabola	ReLU-Sparabola	Sparabola-ReLU	Sigmoid
Starting weights	$w = -1, b = 0$			
Setup results	$w = -3.994$, $b = 6.246$	$w = -3.993$, $b = 6.246$	$w = -0.999$, $b = 0.999$	$w = -18.328$, $b = 8.688$
p	0.5	0.5	0.5	—
β	-1.5	-1.5	0	—
α	—	—	—	0.5
Average deviation	0.001	0.001	0.0001	0.010
Training time, ms	219	218	218	2719
Time to implement 10000000 data table cycles, ms	220	219	183	2843

3.2. Settings of activation functions for logical bases “AND-NOT”, “OR-NOT”

Table 7–12 presents the results of the implementation of the alternative full basis “AND-NOT” and “OR-NOT” using one and two neurons.

Table 7

Characteristics of the setting for the “AND-NOT” function on one neuron

Characteristics of training	S-parabola	ReLU-Sparabola	Sparabola-ReLU	Sigmoid
Starting weights	—	$w_1 = 1, w_2 = 1, b = -1.5$		
Setup results	$w_1 = -1.040$, $w_2 = -1.035$, $b = 0.515$	$w_1 = -0.585$, $w_2 = -0.546$, $b = 1.871$	$w_1 = -0.510$, $w_2 = -0.562$, $b = 3.177$	$w_1 = -16.373$, $w_2 = -16.405$, $b = 24.852$
p	0.9	1	4	—
β	1.75	-0.75	-2	—
α	—	—	—	0.5
Average deviation	0.286	0.186	0.220	0.010
Training time, ms	1198	5064	8801	3223
Time to implement 10000000 data table cycles, ms	692	655	581	1043

Table 8

Characteristics of setting for the “AND-NOT” function on two neurons (Fig. 1(d))

Characteristics of training	S-parabola	ReLU-Sparabola	Sparabola-ReLU	Sigmoid
Starting weights	$w_{11} = -7, w_{21} = -7, w_{12} = -7, w_{22} = -4, w_{12}^* = -11$ $w_{01} = -2.6, w_{02} = 10$			—
Setup results	$w_{11} = -13.998$, $w_{21} = -12.525$, $w_{12} = -1.530$, $w_{22} = -1.461$, $w_{12}^* = -0.326$, $w_{01} = 1.289$, $w_{02} = 1.353$	$w_{11} = -6.892$, $w_{21} = -6.691$, $w_{12} = 0.364$, $w_{22} = 0.341$, $w_{12}^* = 0.193$, $w_{01} = 7.415$, $w_{02} = 0.469$	$w_{11} = -12.926$, $w_{21} = -12.057$, $w_{12} = -1.470$, $w_{22} = -1.431$, $w_{12}^* = -0.297$, $w_{01} = 1.583$, $w_{02} = 1.463$	$w_{11} = -0.073$, $w_{21} = -0.593$, $w_{12} = -4.991$, $w_{22} = -4.559$, $w_{12}^* = 3.812$, $w_{01} = 0.559$, $w_{02} = 5.303$
p	0.5	0.5	0.5	—
β	0.01	0.01	0.01	—
α	—	—	—	1.0
Average deviation	0.009	0.006	0.009	0.039
Training time, ms	538	450	506	10311
Time to implement 10000000 data table cycles, ms	39200	32600	36700	4303800

Table 9

Results of the implementation of the “AND-NOT” function on two neurons (Fig. 1(c))

x_1	x_2	y			
		S-parabola	ReLU-Sparabola	Sparabola-ReLU	Sigmoid
0	0	1.072	1.310	1.621	0.987
1	1	0.107	0.142	0.066	0.010
0	1	0.818	0.722	0.940	0.985
1	0	0.844	0.593	0.940	0.985

Table 10

Comparative characteristics of setting for the “OR-NOT” function on one neuron

Characteristics of training	S-parabola	ReLU-Sparabola	Sparabola-ReLU	Sigmoid
Starting weights	—	$w_1 = 1, w_2 = 1,$ $b = -0.5$	—	$w_1 = 1, w_2 = 1,$ $b = -0.5$
Setup results	$w_1 = -0.135,$ $w_2 = -0.135,$ $b = 0.158$	$w_1 = -0.325,$ $w_2 = -0.420,$ $b = 0.870$	$w_1 = -0.448,$ $w_2 = -0.515,$ $b = 1.583$	$w_1 = -16.226,$ $w_2 = -16.238,$ $b = 7.139$
p	4.0	2.0	3.0	—
β	-0.25	-1.0	-0.75	—
α	—	—	—	0.5
Average deviation	0.296	0.231	0.215	0.010
Training time, ms	5800	2232	6155	1597
Time to implement 10000000 data table cycles, ms	647	677	614	1120

Table 11

Characteristics of setting for the “OR-NOT” function on two neurons (Fig. 1(d))

Characteristics of training	S-parabola	ReLU-Sparabola	Sparabola-ReLU	Sigmoid
Starting weights	$w_{11} = -7, w_{21} = -7, w_{12} = -7, w_{22} = -4, w_{12}{}^* = -11$ $w_{01} = -2.6, w_{02} = 10$			—
Setup results	$w_{11} = -12.977,$ $w_{21} = -12.665,$ $w_{12} = 0.511,$ $w_{22} = 0.496,$ $w_{12}{}^* = 0.327,$ $w_{01} = 1.307,$ $w_{02} = 0.603$	$w_{11} = -6.932,$ $w_{21} = -6.747,$ $w_{12} = -1.395,$ $w_{22} = -1.375,$ $w_{12}{}^* = -0.204,$ $w_{01} = 7.709,$ $w_{02} = 1.558$	$w_{11} = -12.441,$ $w_{21} = -12.075,$ $w_{12} = 0.466,$ $w_{22} = 0.449,$ $w_{12}{}^* = 0.301,$ $w_{01} = 1.530,$ $w_{02} = 0.526$	$w_{11} = 0.508,$ $w_{21} = 0.536,$ $w_{12} = -9.564,$ $w_{22} = -9.557,$ $w_{12}{}^* = -0.360,$ $w_{01} = -0.464,$ $w_{02} = 3.900$
p	0.5	0.5	0.5	—
β	0.01	0.01	0.01	—
α	—	—	—	0.5
Average deviation	0.010	0.007	0.010	0.037
Training time, ms	377	819	279	18819
Time to implement 10000000 data table cycles, ms	1536	1488	1456	2512

Table 12

Results of the implementation of the "OR-NOT" function on two neurons (Fig. 1(c))

x_1	x_2	y			
		S-parabola	ReLU-Sparabola	Sparabola-ReLU	Sigmoid
0	0	0.852	0.681	0.874	0.983
1	1	-0.240	-0.236	-0.452	0.008
0	1	0.178	0.209	0.271	0.009
1	0	0.107	0.218	0.279	0.009

The tables show that when implementing a logical basis on two neurons using parabolic activation functions, the tuning results exceed the results of implementation on a sigmoid in accuracy and speed. Sigmoid is worst tuned to the "AND-NOT" function. When implemented on one neuron, sigmoid works more accurately, although slower.

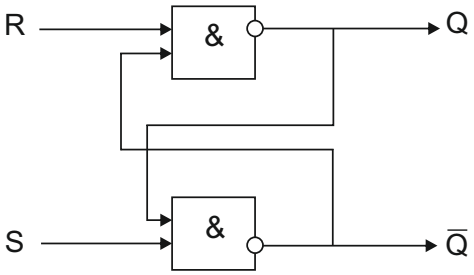
3.3. Tuning neurons to the "XOR" function

To expand the research, new activation functions were additionally configured using the backpropagation method on the "XOR" function using the ANN constructed according to the scheme in Fig. 1(d). The results are shown in Table 13.

Table 13

Characteristics of setting for the "XOR" function on two neurons (Fig. 1(d))

Characteristics of training	S-parabola	ReLU-Sparabola	Sparabola-ReLU	Sigmoid
Starting weights	$w_{11} = -7$, $w_{21} = -7$, $w_{12} = -7$, $w_{22} = -4$, $w_{12}^* = -11$ $w_{01} = -2.6$, $w_{02} = 10$			
Setup results	$w_{11} = -14.176$, $w_{21} = -13.494$, $w_{12} = -2.025$, $w_{22} = -1.963$, $w_{12}^* = -0.633$, $w_{01} = 1.375$, $w_{02} = 0.748$	$w_{11} = -6.872$, $w_{21} = -6.686$, $w_{12} = 1.733$, $w_{22} = 1.691$, $w_{12}^* = 0.392$, $w_{01} = 7.547$, $w_{02} = -1.079$	$w_{11} = -13.496$, $w_{21} = -12.694$, $w_{12} = -1.946$, $w_{22} = -1.876$, $w_{12}^* = -0.586$, $w_{01} = 1.557$, $w_{02} = 0.918$	$w_{11} = -6.999$, $w_{21} = -6.999$, $w_{12} = -6.275$, $w_{22} = -6.154$, $w_{12}^* = -14.208$, $w_{01} = 2.613$, $w_{02} = 9.382$
p	0.5	0.5	0.5	—
β	0.01	0.01	0.01	—
α	—	—	—	1.0
Average deviation	0.009	0.010	0.008	0.035
Training time, ms	603	542	550	23681
Time to implement 10000000 data table cycles, ms	1944	1511	1474	2489

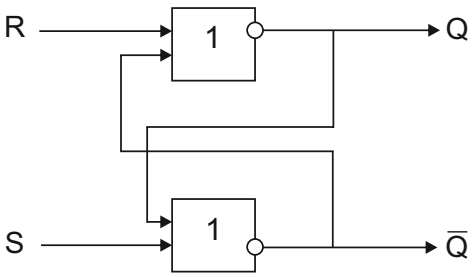


a) RS trigger on “AND-NOT” logic circuits

R	S	Q	$\overline{Q}$	Comment
0	0	1	1	Prohibition
0	1	1	0	Record “1”
1	0	0	1	Record “0”
1	1	×	×	Storage

b) Truth table

Figure 2. Trigger on “AND-NOT” elements



a) RS trigger on “OR-NOT” logic circuits

R	S	Q	$\overline{Q}$	Comment
0	0	×	×	Storage
0	1	1	0	Record “1”
1	0	0	1	Record “0”
1	1	0	0	Prohibition

b) Truth table

Figure 3. Trigger on “OR-NOT” elements

It is evident that the main trends identified in the compilation of logical bases are also preserved in the implementation of a more complex “XOR” function, which is widely used in computer elements.

3.4. Construction of CD elements on logical bases

The CD circuits are implemented by covering them with neurons and neural networks configured on the logical bases “AND”–“OR”–“NOT”, “AND-NOT”, “OR-NOT” or with the sequential formation of more complex elements. Fig. 2 and Fig. 3 show the results of such coverage of the RS-trigger circuits with logical elements on the neurons “AND-NOT” and “OR-NOT”.

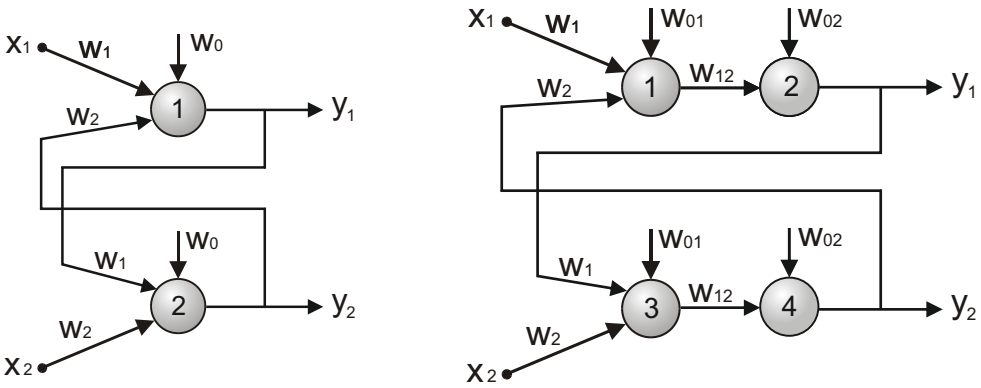
Fig. 4a shows the circuit diagrams of some basic elements of the CD, implemented in the logical basis of “OR-NOT”, and Fig. 4b shows with the addition of the “single jump” element.

Fig. 5a shows the implementation of the RS trigger on two “AND-NOT” neurons, and Fig. 5b adds a “single jump” type element to the outputs, which allows us to obtain clear values of “1” and “0” at the outputs.

The results of the RS-trigger simulation are presented in Tables 16-17.

Let us consider the construction of a neural network for the implementation of a half adder. A single-bit adder (half adder), designated as SM/2, does not have an input carry, since it is the least significant bit of a multi-bit adder. A half adder can be built on the basis of XOR logic circuits and the AND circuit (Fig. 6-7). The logic of SM/2 operation is determined by the truth table.

Based on two half adders and an “OR” circuit, a full one-bit adder SM can be constructed as shown in Fig.8-9.



a) using the outputs of "AND-NOT" neurons

b) adding neurons with the activation function "single jump"

Figure 4. RS trigger in which the "AND-NOT" element is implemented on one neuron

Table 14

Results of the RS trigger operation, in which the "AND-NOT" element is implemented on one neuron

x_1	x_2	y_1	y_2
0	0	Prohibition	
1	1	Storage	
0	1	2.713	-0.702
1	0	0.074	2.638

Table 15

Results of the RS trigger operation, in which the "AND-NOT" element is implemented on one neuron with the addition of neurons with the "single jump" activation function

x_1	x_2	y_1	y_2
0	0	Prohibition	
1	1	Storage	
0	1	1	0
1	0	0	1

Table 16

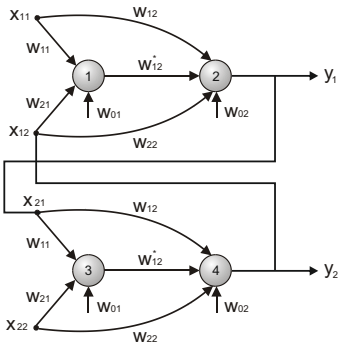
Results of the RS trigger operation, in which the "AND-NOT" element is implemented on two neurons

x_1	x_2	y_1	y_2
0	0	Prohibition	
1	1	Storage	
0	1	0.999	-0.059
1	0	-0.061	0.992

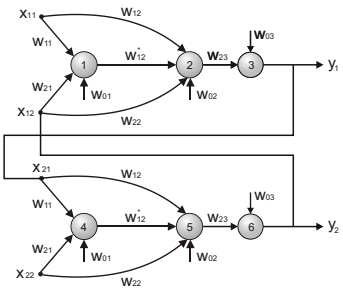
Table 17

Results of the RS trigger operation, in which the "AND-NOT" element is implemented on two neurons with the addition of neurons with the "single jump" activation function

x_1	x_2	y_1	y_2
0	0	Prohibition	
1	1	Storage	
0	1	1	0
1	0	0	1

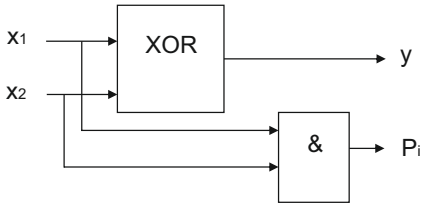


a) using “AND-NOT” neuron outputs



b) adding neurons with the activation function “single jump”

Figure 5. RS trigger in which the “AND-NOT” element is implemented on two neurons

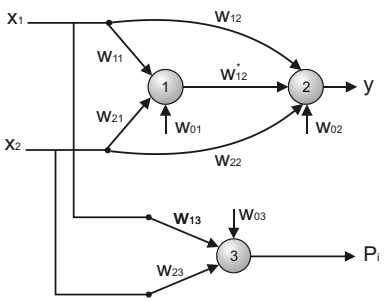


a) Half adder circuit (SM/2)

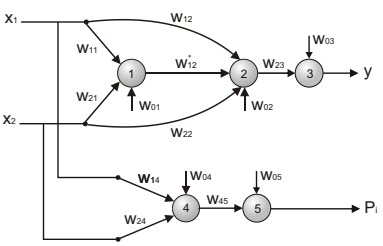
x_1	x_2	y	P_i
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

b) Truth table

Figure 6. Logical diagram of the half adder (SM/2)

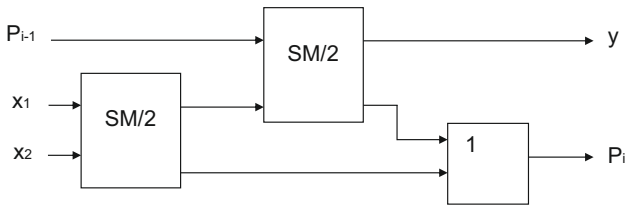


a) base model



b) model with additional neurons (with the “single jump” function)

Figure 7. Models of a half-adder (SM/2) on neurons



a) Full single-bit adder (SM)

x_1	x_2	P_{i-1}	y	P_i
0	0	0	0	0
0	0	1	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

b) SM operation logic

Figure 8. Logic diagram of the sub adder (SM)

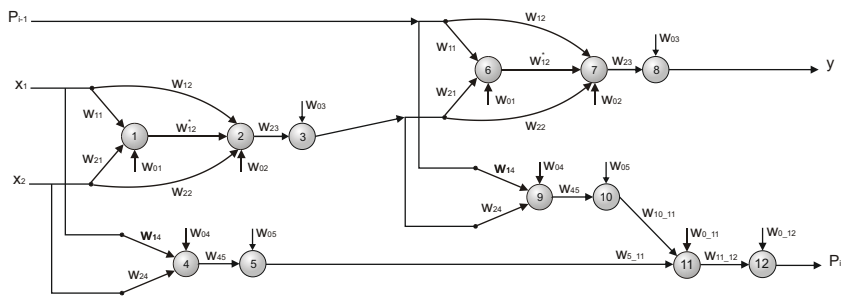


Figure 9. Model of a full adder (SM) on neurons

4. Discussion

Comparison of the obtained results with existing works in the field of implementation of activation functions shows that of interest is not only the decomposition of a complex activation function, for example, a sigmoid, into simpler functions in order to speed up its implementation [1, 2], but also a direct replacement of the function with other functions, for example, an s-parabola and its variations, which allow a more efficient implementation in time. In this case, the effect of acceleration is ensured by several times with some loss in the accuracy of the implementation of thresholds [0,1]. Undoubtedly, of interest is also some slight complication of the VT schemes due to the addition of neurons with the traditional single-jump function to the final cascades, as shown in paragraph 3.4.

Another aspect of acceleration is related to the efficient hardware implementation of activation functions. The special organization of “fast” neurons and neural networks is the subject of, for example, the works [14–16].

Great hopes are placed on bit-parallel circuits for calculating functions included in activation functions. We note a series of works aimed at increasing the speed of solving various problems of comparison, establishing correspondence between streams and performing arithmetic operations on bit vectors, including simultaneous processing of matrix columns [17–20].

In this case, the expected success is associated with the transition to CORDIC algorithms [21–23], which allow regulating the speed and accuracy of calculations. This direction will be developed in subsequent works by the authors.

5. Conclusions

Summing up, we can recapitulate.

1. The composition of activation functions based on the parabola has been expanded, including S-parabola, ReLU-Sparabola and Sparabola-ReLU.
2. Activation functions have been constructed that implement complete logical bases “OR-AND-NOT” and “AND-NOT, OR-NOT” on neurons and neural networks with new activation functions. Compared to implementations based on the “Sigmoid” function, in the general case, the implementation is accelerated by 1.6-1.8 times with a small possible loss in accuracy and setup time. Setup time is not a decisive factor here, since after setup, the neurons are further used without changing the established coefficients, which are fixed.
3. Typical elements of computing equipment have been implemented on them: triggers, half adders, adders.
4. It is planned to implement the s-parabola activation function in multilayer ANNs, including convolutional neural networks of the YOLO class. 5. The possibilities and applications of new activation functions in convolutional neural networks will be investigated.
5. To speed up the calculation of activation functions, bit-parallel calculation schemes will be proposed using the CORDIC family of algorithms.

Author Contributions: Conceptualization, Khachumov M. and Khachumov V.; methodology, Khachumov M.; software, Emelyanova Y.; validation, Khachumov M., Emelyanova Y. and Khachumov V; formal analysis, Emelyanova Y.; investigation, Khachumov M.; resources, Emelyanova Y.; data curation, Emelyanova Y.; writing—original draft preparation, Emelyanova Y.; writing—review and editing, Khachumov M., Khachumov V; visualization, Emelyanova Y.; supervision, Khachumov M.; project administration, Khachumov M.; funding acquisition, Khachumov M. All authors have read and agreed to the published version of the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Russian Science Foundation, grant number 25-21-00222 (https://rscf.ru/en/project/25-21-00222/).

Data Availability Statement: Data sharing is not applicable.

Conflicts of Interest: Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Declaration on Generative AI: The authors have not employed any Generative AI tools.

References

1. Limonova, E., Nikolaev, D. & Alfonso, D. Bipolar morphological neural networks: Gate-efficient architecture for confined environment, 573–580. doi:10.1109/NAECON46414.2019.9058018 (2019).
2. Limonova, E., Nikolaev, D. & Arlazarov, V. Bipolar morphological U-Net for document binarization, 1–9. doi:10.1117/12.2587174 (2021).
3. Limonova, E., Nikolaev, D., Alfonso, D. & Arlazarov, V. ResNet-like architecture with low hardware requirements, 6204–6211. doi:10.1109/ICPR48806.2021.9413186 (2021).

4. Dubey, S., Singh, S. & Chaudhuri, B. Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark. *Neurocomputing* **503,** 1–18. doi:10.1016/j.neucom.2022.06.111 (2022).
5. Feng, J. & Lu, S. Performance Analysis of Various Activation Functions in Artificial Neural Networks. *Journal of Physics Conference Series,* 1–7. doi:10.1088/1742-6596/1237/2/02203 (2019).
6. Akgül, I. Activation functions used in artificial neural networks. *In book: Academic Studies in Engineering,* 41–58 (Oct. 2023).
7. Arce, F., Zamora, E., Humberto, S. & Barrón, R. Differential evolution training algorithm for dendrite morphological neural networks. *Applied Soft Computing* **68,** 303–313. doi:10.1016/j.asoc.2018.03.033 (2018).
8. Dimitriadis, N. & Maragos, P. Advances in the training, pruning and enforcement of shape constraints of morphological neural networks using tropical algebra, 3825–3829. doi:10.48550/arXiv.2011.07643 (2021).
9. Limonova, E., Nikolaev, D., Alfonso, D. & Arlazarov, V. Bipolar morphological neural networks: gate-efficient architecture for computer vision. *IEEE Access* **9,** 97569–97581. doi:10.1109/ACCESS.2021.3094484 (2021).
10. Galushkin, A., Sudarikov, V. & Shabanov, E. Neuromathematics: the methods of solving problems on neurocomputers. **2,** 1179–1188. doi:10.1109/RNNS.1992.268515 (1992).
11. Khachumov, M. & Emelyanova, Y. Parabola as an Activation Function of Artificial Neural Networks. *Scientific and Technical Information Processing* **51,** 471–477. doi:10 . 3103 / S0147688224700382 (2024).
12. Khachumov, M., Emelyanova, Y. & Khachumov, V. Parabola-Based Artificial Neural Network Activation Functions, 249–254. doi:10.1109/RusAutoCon58002.2023.10272855 (Sept. 2023).
13. Hanche-Olsen, H. & Holden, H. The Kolmogorov–Riesz compactness theorem. **28,** 385–394. doi:10.1016/j.exmath.2010.03.001 (June 2009).
14. Nabavinejad, S., Reda, S. & Ebrahimi, M. Coordinated Batching and DVFS for DNN Inference on GPU Accelerators. **33,** 1–12. doi:10.1109/TPDS.2022.3144614 (Oct. 2022).
15. Trusov, A., Limonova, E., Slugin, D., Nikolaev, D. & Arlazarov, V. Fast Implementation of 4-bit Convolutional Neural Networks for Mobile Devices, 9897–9903. doi:10.1109/ICPR48806.2021.9412841 (Sept. 2020).
16. Shashev, D. & Shatravin, V. Implementation of the sigmoid activation function using the reconfigurable computing environments. Russian. *Tomsk State University Journal of Control and Computer Science,* 117–127. doi:10.17223/19988605/61/12 (2022).
17. Hyyro, H. & Navarro, G. Bit-Parallel Computation of Local Similarity Score Matrices with Unitary Weights. *International Journal of Foundations of Computer Science.* doi:10.1142/S0129054106004443.
18. Hyyro, H. Explaining and extending the bit-parallel approximate string matching algorithm of Myers (2001).
19. Hyyro, H. & Navarro, G. Faster bit-parallel approximate string matching, 203–224 (2002).
20. Hyyro, H. & Navarro, G. Bit-parallel witnesses and their applications to approximate string matching. *Algorithmica* **41,** 203–231 (2005).
21. Zakharov, A. & Khachumov, V. Bit-parallel Representation of Activation Functions for Fast Neural Networks. **2,** 568–571 (2014).
22. Volder, J. The Birth of Cordic. *The Journal of VLSI Signal Processing-Systems for Signal, Image, and Video Technology* **25,** 101–105. doi:10.1023/A:1008110704586 (2000).
23. Chetana & Sharmila, K. VLSI Implementation of Coordinate Rotation Based Design Methodology using Verilog HDL. doi:10.1109/ICAIS56108.2023.10073928 (2023).

Information about the authors

Khachumov, Mihail V.—Candidate of Physical and Mathematical Sciences, Senior Researcher at Federal Research Center "Computer Science and Control" of Russian Academy of Sciences (e-mail: khmike@inbox.ru, phone: +7(926)7104291, ORCID: 0000-0001-5117-384X, Scopus Author ID: 55570238100)

Khachumov, Vyacheslav M.—Doctor of Technical Sciences, Chief Researcher at Federal Research Center "Computer Science and Control" of Russian Academy of Sciences (e-mail: vmh48@mail.ru, ORCID: 0000-0001-9577-1438, Scopus Author ID: 56042383100)

Emelyanova, Yuliya G.—Candidate of Technical Sciences, Senior Researcher at Ailamazyan Program Systems Institute of RAS (e-mail: yuliya.emelyanowa2015@yandex.ru, ORCID: 0000-0001-7735-6820, Scopus Author ID: 57202835704)

УДК 004.032.26
PACS 07.05.Mh, 07.05.Kf
DOI: 10.22363/2658-4670-2025-33-3-242-259 EDN: HFFBMV

Построение и моделирование работы элементов вычислительной техники на быстрых нейронах

М. В. Хачумов[1, 2], Ю. Г. Емельянова[3], В. М. Хачумов[1, 2, 3]

[1] Российский университет дружбы народов, ул. Миклухо-Маклая, д.6, Москва, Россия, 117198

[2] Федеральный исследовательский центр «Информатика и управление» Российской академии наук, ул. Вавилова, д.44, кор.2, Москва, Россия, 119333

[3] ИПС им. А. К. Айламазяна РАН, ул. Петра Первого, д.4а, Переславль-Залесский, Ярославская область, Россия, 152021

Аннотация. Статья посвящена построению быстрых нейронов и нейронных сетей для реализации двух полных логических базисов и моделирования на их основе устройств вычислительной техники. Основная идея заключается в формировании быстрой функции активации на основе полупарабол и её вариаций, имеющих эффективную вычислительную поддержку. Построенные функции активации отвечают основным требованиям, позволяющим настраивать логические схемы методом обратного распространения ошибки. Основным результатом является получение полных логических базисов, открывающих путь к построению произвольных логических функций. Представлены и протестированы модели таких элементов как триггер, полусумматор, сумматор, составляющих основу различных конкретных вычислительных устройств. Показано, что новые функции активации позволяют получать быстрые решения при небольшом снижении качества по сравнению с эталонными выходами. Для стандартизации выходов предлагается комбинировать построенные схемы с функцией активации типа единичный скачок.

Ключевые слова: новые функции активации, парабола, полный логический базис, модели элементов, быстродействие, экспериментальные исследования