IEEE Second International Conference on Data Stream Mining & Processing
August 21-25, 2018, Lviv, Ukraine

Neural-Like Means for Data Streams Encryption and Decryption in Real Time

Ivan Tsmots
Lviv Polytechnic National University
Lviv, Ukraine
ivan.tsmots@gmail.com

Viktor Khavalko
Lviv Polytechnic National University
Lviv, Ukraine
khavalkov@gmail.com

Yurii Tsymbal
Lviv Polytechnic National University
Lviv, Ukraine
yurij.tsymbal@gmail.com

Oleksa Skorokhoda
Lviv Polytechnic National University
Lviv, Ukraine
oskorokhoda@lp.edu.ua

Taras Tesluyk
Lviv Polytechnic National University
Lviv, Ukraine
taras.teslyuk@gmail.com

***Abstract* — The "model of successive geometric transformations" paradigm has been adapted for the implementation of parallel-streaming neural network encryption-decryption of data in real time. A model and structure of a parallel-streaming neural-like element for the mode have been developed.**

Keywords—intensive data stream; neural networks; geometric transformations model

I. INTRODUCTION

The latest information technologies are becoming global in the modern world. Their development and development of communications provides ever-widening opportunities for access to information resources and the transfer of large amounts of data for unlimited distances. In the context of the intensive development of the market for information products and services, information has become a full-fledged product that has its own consumer properties and cost characteristics.

The widespread introduction of information technology makes a relevant problem for the protection of the transmission of information using cryptographic methods that provide encryption of the ready-to-transmit information. The encrypted information is transmitted by a communication channel to an authorized user, who, after receiving it, performs decryption using a reverse transformation. Cryptographic transformations are carried out using special algorithms. In order to encrypt and decrypt real-time data streams, it is suggested to use neural-like network algorithms, the key in which is the network architecture, weighting factors, and masking codes.

Real-time encryption and decryption of intensive streams can be ensured through VLSI-implementation of the corresponding algorithms. For the synthesis of neural-like elements and neural-like real-time networks, it is necessary to develop new parallel-stream neural elements and networks that provide spatiotemporal parallelization of encryption and decryption algorithms. To implement such algorithms programmable logic integrated circuits (FPGAs) can be used. The main advantages of FPGA are low cost, affordability, high performance, reliability and availability of a variety of well-developed and efficient software tools for automated design. A significant feature of the latest generation of FPGAs is the possibility of reconfiguration of such circuits during operation.

Therefore, development of hardware neural-like networks for encryption-decryption of intense data streams in real time is the actual problem.

II. ANALYSIS OF PUBLICATIONS

Analysis of works [1-15] provided features highlighting of real-time neurocomputers tasks and architectures. Real-time encryption-decryption tasks are characterized by high intensity, continuity of incoming data streams and increased requirements for key lifetime.

The works [3-9] analyze the existing neural networks and the means of their implementation. The analysis showed that the vast majority of neural networks are implemented by software. Such neural networks have relatively low performance and do not provide real-time for the processing of intense data streams.

From the analysis of works [1-13] it is evident that in order to ensure the high performance of neural networks in real time, it is necessary to use hardware implementation and a modern element base. A prerequisite for the construction of efficient neurocomputer architectures is a usage of three basic principles: the parallelism of processing; programmability of the structure; regularity (uniformity) of the structure. This approach is provided by a combination of principles of conveyance, vector and matrix software and hardware organization of computing on the basis of the latest elemental base technologies. These three principles correspond to the problem of adequately mapping the spatial-temporal algorithmic structure of computational processes into the architecture of parallel computers [1].

Several directions in the creation of a new approach to computing processes in neurocomputers were identified in works [1-15]: high level of paralleling at all levels of data processing; pipeline principle of data processing; organization of high-level internal language with hardware support; increasing the regularity of hardware implementations, for example, using the systolic structures; use of multicast access storage environment; the transition to a universal matrix-algebraic system instead of the algebra of

978-1-5386-2874-4/18/$31.00 ©2018 IEEE

real numbers; hardware implementation of basic operations in computational procedures.

The main components of the hardware neural network are artificial neural elements. There are different models of an artificial neuron. Choosing an artificial neuron model depends on the requirements of specific applications. In [10], the models and VLSI structures of the parallel-vertical type formal neuron are reviewed, which differ in the way of receipting and processing of input data and weight coefficients - using multiplexing of tires, combining the processes of data receipt and processing, and with table formations of macro-partial results. The disadvantage of these models and VLSI structures of the formal neuron is relatively low performance.

It follows from the analysis that the synthesis of the hardware neural network for real-time data encryption and decryption tasks requires the development of a new model and new VLSI structures of the neural element, which should be focused on the processing of intensive data streams.

One of the promising directions for the construction of high-performance neural network means is the application of the "model of successive geometric transformations" (MSGT) paradigm proposed and developed by R. Tkachenko [16-17].

An alternative approach to solving data encryption tasks using algebraic transformations is considered in [18-19].

The purpose of this paper is to adapt the model of successive geometric transformations to the tasks of data encryption and decryption, development of a model and a structure of a parallel-streaming neural-like element and synthesis on its base of a parallel-streaming neural-like network for data streams encryption and decryption.

III. MAIN PART

A. The model of successive geometric transformations

The basis of this paradigm is the non-iterative approach to the teaching of a neural-like network, which involves the direct calculation of weight coefficients during the gradual reduction of the dimension of the space of incoming multidimensional data on neurons in the hidden layer [16]. In this case, the representation of incoming multidimensional data in the new orthogonal basis is used, executed on the basis of the non-iterative greedy algorithm of the most distant point.

The hardware implementation of such neural-like networks using VLSI structures can be greatly simplified if the input, output and weighting coefficients of the MSGT network are presented in a fixed-point format. For this, a preliminary scaling of the input data is anticipated.

When choosing the structure of a neural-like network for real-time encryption-decryption of data streams, it is proposed to use the architecture of an auto-associative network with one hidden layer [17] (Fig. 1). Successive geometric transformations are depicted by lateral connections between neurons in a hidden layer.

This network structure is quite versatile and can be used to solve various tasks that involve converting incoming data and its subsequent recovering: encoding input data for compression, block symmetric encryption, overlaying of digital watermarks (on image) and steganography. Encryption and decryption modules, respectively, form and use a key that forms the parameters of the trained neural network. To improve the cryptographic stability of the hidden layer output, the structure of the network can be supplemented with a block of encryption using a one-time notepad (masking by XOR).

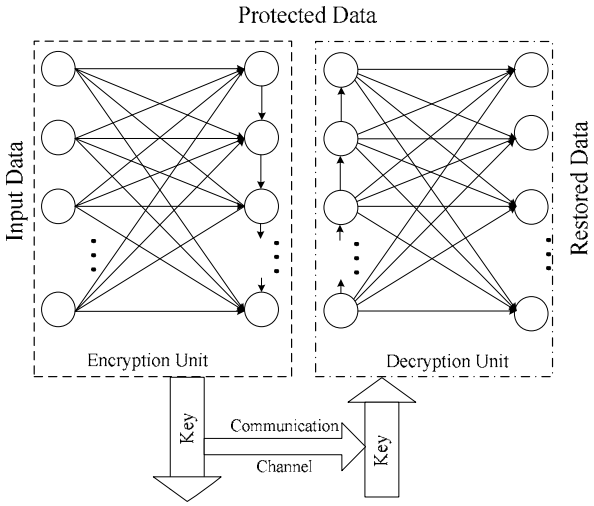

Fig. 1. The graph of a neural-like network for encryption-decryption of data based on a model of successive geometric transformations

With the number of neurons in the hidden layer equal to the number of input and output neurons, it is possible to restore without loss secure data on the network output that will be obtained at the outputs of the hidden layer neurons. Reverse consecutive geometric transformations between hidden and output neurons are used to restore data on network outputs.

It is possible to serially connect several encryption blocks and subsequent appropriate decryption units to create a cascading multilayer network.

Performance of the encryption module can be described as follows:

$$h = F(X, key), \tag{1}$$

where X, h – signals on the input and output of the module, respectively; $key = \{N, W, mask\}$ - a key that consists of a given number of neurons in the input (hidden, output) layer of the network N, the matrix of weight coefficients W, and the mask for a one-time notebook $mask$, F - the function that specifies straight consequent geometric transformations.

Then the functioning of the decryption module is described as:

$$\tilde{X} = \overline{F}(h, key), \tag{2}$$

where h – the signal at the input of the module (protected data), $\tilde{X}$ – the signal at the output of the module (recovered data), $\overline{F}$ – the function that specifies reverse consecutive geometric transformations.

The lifetime of a key key then depends on:

- the number of neurons N ;
- the bit-width n of the mask;
- the number of serially connected encryption and decryption blocks in the cascade network *k.*

A. The model of a parallel-streaming neural-like element

The main components on the basis of which parallel-streaming neural systems are synthesized are neural elements. The peculiarity of the work of neural-like systems is that they are focused on solving specific problems using the "model of consecutive geometric transformations" paradigm. In a parallel-streaming neural-like element, which is realized on the basis of such a paradigm, weight coefficients W_j are pre-calculated and do not change or change very rarely in the process of operation. In the general case, the neural element makes a transformation in accordance with the formula:

$$Y = f(\sum_{j=1}^{N} W_j X_j), \tag{3}$$

where Y – the output signal of the neural element, f – the activation function, N – the number of inputs.

From the formula (3) it follows that the processing of data in neural elements is reduced to calculations of scalar product and activation functions *f*. During calculation of the scalar product in a parallel-streaming neural-like element, weighted coefficients W_j are pre-computed and stored in memory, and the input data arrives at the same time on all inputs as the parallel binary code.

In parallel-streaming neural-like systems, which are oriented on VLSI implementation, it is expedient to use table-algorithmic methods and the basis of elementary operations for their implementation. The basic operation of a parallel-streaming neural-like element is the operation of calculating the scalar product, which for VLSI-implementation must be provided on the basis of elementary operations. To calculate the scalar product on the basis of elementary operations, it is expedient to use multiplication algorithms with the direct formation of partial products, since they are regular and well-structured. The most common of these are algorithms of multiplication with the analysis of a one bit-slice. The multiplication of the numbers given in the binary complement code, with the analysis of a single digit of the multiplier, is written as follows:

$$\begin{aligned} C_j = W_j X_j &= \sum_{i=0}^{n-1} (-1)^{2^i} 2^{-i} W_j x_i \\ &= \sum_{i=0}^{n-1} (-1)^{2^i} 2^{-i} P_{ji} \end{aligned}, \tag{4}$$

where n – the bit-length of the multiplier; x_i – the value of the i-th bit of the multiplier; P_{ji} – the i-th partial product.

Using the multiplication algorithm (4) we develop a parallel-flow algorithm for scalar products calculation. The algorithm for calculating a scalar product using the multiplication algorithm (4) is written as follows:

$$\begin{aligned} Z = \sum_{j=1}^{N} W_j X_j &= \sum_{j=1}^{N} \sum_{i=0}^{n-1} (-1)^{2^{-i}} 2^{-i} P_{ji} \\ &= \sum_{i=0}^{n-1} (-1)^{2^{-i}} 2^{-i} \sum_{j=1}^{N} P_{ji} \end{aligned}. \tag{5}$$

If in the formula (5) the sum of i-th partial products $\sum_{j-1}^{N} P_{ji}$ replace by the i-th macro-partial product P_{Mi} then we get

$$Z = \sum_{j=1}^{N} W_j X_j = \sum_{i=1}^{n} (-1)^{2^{-i}} 2^{-i} P_{Mi} \tag{6}$$

In neural-like elements, weighted coefficients W_j are pre-calculated, that is, they can be considered as constants. In this case, when calculating the scalar product by formula (6), we can calculate the table of macro-partial products P_{Mi} in advance. The calculation of the values of the table of macro-partial products P_{Mi} is carried out according to the following formula:

$$P_{Mi} = \begin{cases} 0, \quad if \quad x_{1i} = x_{2i} = x_{3i} = ... = x_{Ni} = 0 \\ W_1, \quad if \quad x_{1i} = 1, \quad x_{2i} = x_{3i} = ... = x_{Ni} = 0 \\ W_2, \quad if \quad x_{1i} = 0, x_{2i} = 1, x_{3i} = ... = x_{Ni} = 0 \\ W_1 + W_2, \quad if \quad x_{1i} = 1, x_{2i} = 1, x_{3i} = ... = x_{Ni} = 0 \\ \vdots \\ W_2 + ... + W_N, \cdots \\ if \quad x_{1i} = 0, x_{2i} = x_{3i} = ... = x_{Ni} = 1 \\ W_1 + W_2 + ... + W_N, \\ if \quad x_{1i} = x_{2i} = x_{3i} = ... = x_{Ni} = 1 \end{cases} \tag{7}$$

The volume of the table of macro-partial products P_{Mi} is determined by the formula $Q = 2^N$.

In the parallel-streaming scalar product calculation by the formulas (6) and (7), the spatiotemporal parallelization of the computation process is used. The peculiarity of the parallel-streaming implementation of scalar product calculations is the use of n identical processing elements (steps) and pipelining of the process. For such implementation, it is advisable to use the algorithm to calculate the scalar product starting with the analysis of least significant bits, which will ensure the use of $(n + \log_2 N)$-bit adders in all processor elements. In this case, in each i-th processor element the following operation will be performed:

$$Z_i = 2^{-1} Z_{i-1} + P_{Mn-(i-1)}, \quad (8)$$

where $Z_0 = 0$.

Analytically, the model of a parallel-streaming neural-like element can be written as follows:

$$Y = f_a \left(\begin{array}{l} f_{3n1}(f_{TP_{Mn}}(f_{\sum_1}(f_{Z_1R1}(...(f_{3ni}(f_{TP_{M(n-i-1)}} \\ (f_{\sum_i}(f_{Z_iR1}(...(f_{3nn}(f_{TP_{M1}}(f_{\sum_n})))))))))))) \end{array} \right) \quad (9)$$

where Y – output of the neural-like element; f_a – activation function; f_{3n1} – buffer memorization of incoming n bit data, $f_{TP_{Mn}}$ – tabular formation of macro-partial products for n-th bits, $f_{\sum_1}$ – calculation of the first partial result $Z_1 = 2^{-1} Z_0 + P_{Mn} = P_{Mn}$, f_{R1Z_1} – shift to the right for one level of the first partial result $R1Z_1$, f_{3ni} – buffer memorization of (n-i-1)-bit operands, $f_{TP_{M(n-i-1)}}$ – tabular formation of macro-partial products for (n-i-1)-th bits of operands, $f_{\sum_i}$ – calculation of i-th partial result $Z_i = 2^{-1} Z_{i-1} + P_{M(n-i-1)}$, f_{R1Z_i} – shift to the right for one level of i-th partial result $R1Z_i$, f_{3nn} – buffer memorization of 1st bits of operands, $f_{TP_{M1}}$ – tabular formation of macro-partial products for 1st bits of operands, $f_{\sum_n}$ – calculations of n-th partial result $Z_n = 2^{-1} Z_{n-1} + P_{M1}$.

The structure of the model of a parallel-streaming neural-like element, which implements the expression (9), is given in Fig. 2

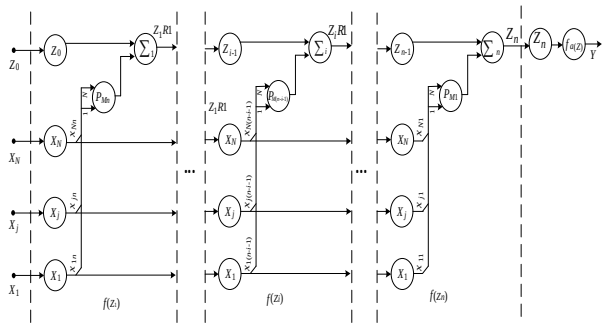

Fig. 2. Model of parallel-streaming neural-like element

The main components of this model are: buffer pipeline memory, memory of macro-partial products and $(n + \log_2 N)$-bits adders. The peculiarity of this model is the combination of simultaneous processing of n input data arrays, which provides high performance.

A. Structure of a parallel-streaming neural-like element

The development of VLSI structures of a parallel-stream neural-like element for real-time neural networks synthesis with high efficiency of equipment use is proposed to carry out on the basis of an integrated approach based on the capabilities of a modern element base, covering the methods, algorithms and structures of hardware of neural networks, taking into account the requirements of specific applications. For a complete use of the advantages of modern VLSI technology, the following principles are proposed on the basis of a parallel-streaming neural-like element:

- use of the basis of elementary arithmetic operations;
- preliminary calculation of weighting factors;
- tabular formation of macro products and activation functions;
- pipeline and spatial parallelism;
- homogeneity and modularity of the structure.

In developing the structure of a parallel-streaming neural-like element, we will use its model (Fig. 2). To switch from a model to a structure of a parallel-streaming neural-like element it is necessary to display all components of the model on hardware. In such a neural-like element, the components of the calculation of the activation function and macro-partial products are implemented using memory tables, buffer memorization - using registers, and the calculation of partial results - on the adders.

The structure of the parallel-streaming neural-like element is shown in Fig. 3, where *PU* – processing unit, *Rg* – register, *RAM* – random access memory, *Add* – adder, P_{Mi} –input of macro-partial products, C_1, C_2 – first and second control inputs, $X_1,\ldots,X_n$ – n data inputs, Wr_1, Wr_2 – first and second input for writhing into RAM.

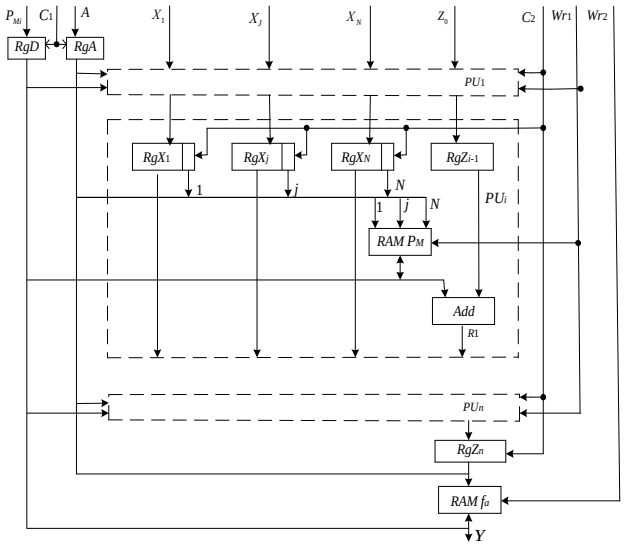

Fig. 3. Structure of a parallel-streaming neural-like element

The calculation of the scalar product in this device requires a preliminary calculation of the weight coefficients W_j and on their basis the formation of the 2^N macro-partial products P_{Mi} according to formula (7), which must be written in the RAM P_M in all PUs. Before beginning the recording of macro products P_{Mi} by signal 0 from the input $C2$ outputs of registers Rg_{X1} - Rg_{XN} in all PUs are switched in the third state and RAMP_M in all PUs are switched to the recording mode using the signal 0 from the input Wr_1. By signaling 1 from input C_1 outputs of registers *RgD* and *RgA*

are set to transmit data and addresses to the corresponding inputs of RAM P_M. Recording of macro-partial products P_{Mi} in the RAM P_M of all PUs is carried out simultaneously starting from the zero address, which is set at the outputs of the registry *RgA*. To write the *k*-th macro-partial product in the RAM P_M ($k = 1, ..., 2^N$), it must be written in the register *RgD*, and in the *RgA* – the value of *k*-1.

The calculation of the activation function in a neural-like element is carried out tabularly. The values of the activation function are recorded in RAM f_a using the registers *RgD* and *RgA*. To write data in RAM f_a it must be switched to recording mode using the signal 0 from the input Wr_2. After the data is recorded in the RAM f_a and RAM P_M, they are switched to read mode, and registers *RgD* and *RgA* are set to the third state.

The characteristic of the developed structure of the parallel-streaming neural-like element (Fig. 3) is its implementation on *n* PU of the same type, which operate according to the pipeline principle and provide the time parallelization of the algorithm of work in such a way that the results of the work of *i*-th PU are input data for (*i*+1)-th PU. The velocity of a parallel-streaming neural-like element is determined by the tact of the pipeline's operation, which is determined by the following formula:

$$T_\varepsilon = \tau_{P\gamma} + \tau_{PAM} + \tau_{A\delta\delta}, \qquad (10)$$

where t_{Rg} – time delay of the register; t_{RAM} – time to read data from memory; t_{Add} – time of addition on the adder.

D. Basic structure of parallel-stream neural network for encryption-decryption of data

The basic structure of the parallel-stream neural network for data encryption-decryption in real time will be synthesized on the basis of the developed parallel-streaming neural-like element (Fig. 3). The purpose of synthesis of such a neural-like network is to obtain a modular and regular structure oriented on VLSI technology.

The initial information for the synthesis of a real-time neural-like network is:

- algorithms of training and operation of the neural network;
- graph representation of the neural network;
- the number of input N and neurons K;
- the intensity of incoming data;
- requirements for the interface;
- bit-length of input data, weighting factors, and accuracy of calculations;
- technical and economic requirements and restrictions.

When synthesizing a neural-like network it is necessary to ensure its functioning in real time with minimal hardware costs. The transition from graph representation to the hardware structure of a neural-like network formally reduces to minimizing hardware costs while providing a real-time mode.

The structure of the neural-like network for data encryption is shown in Fig. 4, where *PU* – processing unit, *Rg* – register, *RAM* – random access memory, *Add* – adder, *Sub* – subtractor, C_1, C_2, C_3 – first, second and third control inputs, *InD* – data input, *OutY* – output of the result of encryption.

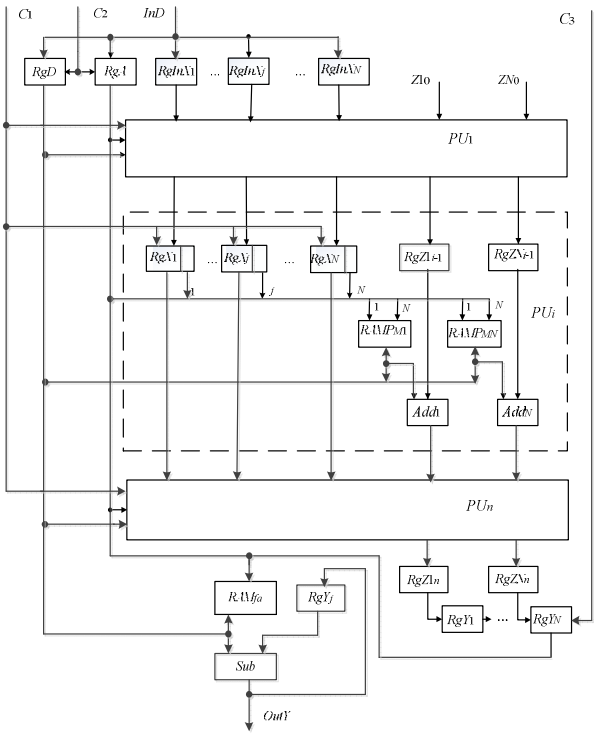

Fig. 4. Structure of a parallel-stream neural network for data streams encryption

Encryption of data streams using a parallel-stream neural network requires a preliminary calculation of weight coefficients W_j, the formation of macro-partial products P_M and their simultaneous recording in all RAM P_M. The peculiarity of the developed structure of the parallel-streaming neural-like encryption network is that the data move consecutively one after another and with the help of inputs $RgIn_1, ..., RgIn_N$ is converted into a parallel stream of data entering the input of the first PU_1. Parallel-streaming neural network is implemented based on *n* identical PU_n, which operates according to the pipeline principle. The operating time of the pipeline of such a network is equal to the tact of the neural-like element (10). In each cycle of work, calculated scalar products are written in registers $RgZ1_n, ..., RgZN_n$, and after them in registers $RgY_1, ..., RgY_N$ with the help of which parallel-sequential transformation of the receipt of scalar products is performed. At the output of the subtractor *Sub* the stream of encrypted data is formed.

IV. CONCLUSIONS

The "model of successive geometric transformations" paradigm has been adapted for the implementation of parallel-streaming neural network encryption-decryption of data in real time.

A model and structure of a parallel-streaming neural-like element have been developed, which provides spatiotemporal parallelization of the process of calculation and its implementation on the basis of *n* processing units of the same type.

For the VLSI-implementation of parallel-streaming neural-like element and network, a table-algorithmic method of calculation on the basis of elementary operations has been used.

Following principles are proposed for implementation of a parallel-streaming neural-like element: use of the basis of elementary arithmetic operations; preliminary calculation of weighting factors; tabular formation of macro products and activation functions; pipeline and spatial parallelism; homogeneity and modularity of the structure.

The velocity of a parallel-streaming neural-like network for data encrypting-decrypting is determined by the tact of the pipeline's operation, which is determined as the sum of the times of data delay on the register, reading data from memory and adding of two numbers.

REFERENCES

[1] A. V. Palagin, and Yu. S. Yakovlev, System integration of computer equipment. Vinnytsia: UNIVERSUM-Vinnytsia, 2005. (in Russian)

[2] V. P. Gribachev, "Element base of hardware implementations of neural networks," in Components and technologies, no. 8, 2006. (in Russian)

[3] S. Haykin, Neural networks and learning machines, 3rd ed. New York: Prentice Hall, 2009.

[4] Ye. V. Bodyanskiy and O. G. Rudenko, Artificial neural networks: architectures, learning, applications. Kharkiv: TELETEH, 2004. (in Russian)

[5] W. S. McCulloch, and W. Pitts, "A logical calculus of the ideas immanent in nervous activity" in The Bulletin of Mathematical Biophysics, vol. 5, iss. 4, pp. 115–133, 1943.

[6] ADALINE (Adaptive linear) [Electronic Resource]: http://www.cs.utsa.edu/~bylander/cs4793/learnsc32.pdf

[7] K. Fukushima, "Cognitron: A self-organizing multilayered neural network" in Biological cybernetics, vol. 20, iss. 3-4, pp. 121–136, 1975.

[8] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities" in Proceedings of the national academy of sciences, vol. 79, iss. 8, pp. 2554–2558, 1982.

[9] J. Cao, and J. Liang, "Boundedness and stability for Cohen–Grossberg neural network with time-varying delays," in Journal of Mathematical Analysis and Applications, vol. 296, iss. 2, pp. 665–685, 2004.

[10] Yu. M. Rashkevich, R. O. Tkachenko, I. G. Tsmots, and D. D. Peleshko, Non-linear methods, algorithms and structures for processing of signals and images in real time: monograph. Lviv: Lviv Polytechnic Publishing House, 2014. (in Ukrainian)

[11] I. G. Tsmots, O. V. Skorokhoda, and B. I. Balych, "Model and VLSI structures of the parallel-vertical type formal neuron using bus multiplexing," in Modeling and Information Technologies, Digest of Scientific Papers of the G.E. Puhov Institute of Modeling Problems in the Energy, Lviv, vol. 67, pp. 160-166, 2013. (in Ukrainian)

[12] I. G. Tsmots, O. V. Skorokhoda, and V. B. Krasovskii, "Models and VLSI structures of a parallel-vertical type formal neuron combining the processes of data incoming and processing," in Modeling and Information Technologies, Digest of Scientific Papers of the G.E. Puhov Institute of Modeling Problems in the Energy, Lviv, vol. 70, pp. 137-145, 2013. (in Ukrainian)

[13] I. G. Tsmots, O. V. Skorokhoda, and B. I. Balych, "Model and VLSI structure of a parallel-vertical type formal neuron with tabular macro-partial results," in Modeling and Information Technologies, Digest of Scientific Papers of the G.E. Puhov Institute of Modeling Problems in the Energy, Lviv, vol. 73, pp. 133-138, 2014. (in Ukrainian)

[14] I. G. Tsmots, O. V. Skorokhoda, and V. M. Tesliuk, A device for calculating scalar product. Patent № 101922 Ukraine, G06F 7/38. Bul. no. 9, 2013. (in Ukrainian)

[15] I. Tsmots, O. Skorokhoda, V. Rabyk, and I. Ignatyev, "Basic vertical-parallel real time neural network components," XIIth International Scientific and Technical Conference "Computer Sciences and Information Technologies" *(CSIT)*, Lviv, pp. 344–347, 2017.

[16] I. Izonin, R. Tkachenko, D. Peleshko, T. Rak, and D. Batyuk, "Learning-based image super-resolution using weight coefficients of synaptic connections," Xth International Scientific and Technical Conference "Computer Sciences and Information Technologies" (CSIT), Lviv, pp. 25-29, 2015.

[17] Y. Tsymbal, and R. Tkachenko, "A digital watermarking scheme based on autoassociative neural networks of the geometric transformations model," 2016 IEEE First International Conference on Data Stream Mining & Processing (DSMP), Lviv, pp. 231-234, 2016.

[18] M. Nazarkevych, R. Oliiarnyk, H. Nazarkevych, O. Kramarenko, and I. Onyshschenko, "The method of encryption based on Ateb-functions," 2016 IEEE First International Conference on Data Stream Mining & Processing (DSMP), Lviv, pp. 129-133, 2016.

[19] I Dronyuk., M. Nazarkevych, and Z. Poplavska, "Gabor filters generalization based on ateb-functions for information security," in Advances in Intelligent Systems and Computing, vol. 659, pp. 195-206, 2018