Parallel Method of Neural Network Synthesis Based on a Modified Genetic Algorithm Application

Serhii Leoshchenko[1[0000-0001-5099-5518]], Andrii Oliinyk[2[0000-0002-6740-6078]], Stepan Skrupsky[3[0000-0002-9437-9095]], Sergey Subbotin[4[0000-0001-5814-8268]] and Tetiana Zaiko[5[0000-0003-1800-8388]]

[1,2,4,5] Dept. of Software Tools, National University "Zaporizhzhia Polytechnic", Zaporizhzhia 69063, Ukraine
[3]Dept. of Computer Systems and networks, Dept. of Software Tools, National University "Zaporizhzhia Polytechnic", Zaporizhzhia 69063, Ukraine
[1]sergleo.zntu@gmail.com, [2]olejnikaa@gmail.com, [3]sskrupsky@gmail.com, [4]subbotin@zntu.edu.ua, [5]nika270202@gmail.com

Abstract. In our days, the using of neural network technologies is relevant in solving applied problems in various fields of science and industry. Such tasks successfully solved by artificial neural networks include: forecasting, classification, as well as diagnostics and detection of pre-emergency situations at hazardous industrial facilities. However, one of the main problems of neural networks implementation is their synthesis: the choosing of topology and setting of the weights (training process). This paper describes a parallel method of neural network synthesis based on a modified genetic algorithm.

Keywords: neural networks, synthesis, sequential, parallel, genetic method.

1 Introduction

One of the most promising areas of application of artificial neural networks (ANN) is industry and industrial applications. In this area, there is a noticeable trend of transition to production modules with a high level of automation, which requires an increase in the number of intelligent self-regulating and self-adjusting machines. However, production processes are characterized by a large variety of dynamically interacting features, which complicates the creation of adequate analytical models. Modern production is constantly becoming more complicated. This slows down the introduction of new technological solutions. In addition, in some cases, successful analytical mathematical models show failure due to lack of computing power [1]. In this regard, there is a growing interest in alternative approaches to modeling of production processes using ANN, providing opportunities to create models that work in real time with small errors, capable of learning more in the process of using. The advantages of ANN make their use attractive for tasks such as:

– prediction and forecasting;

- planning;
- quality management;
- manipulators and robotics control;
- production safety: fault detection and emergency prevention;
- process control: optimization of production processes; monitoring and visualization of dispatching information.

Today, ANN-based forecasting is most fully implemented in finance and the economy. In industry, neural networks can be useful, for example, when creating a model of enterprise risk management [2], planning the production cycle [3]. Modeling and optimization of production is characterized by high complexity, a large number of variables and constants that are not defined for all possible systems. Traditional analytical models can often be constructed only with considerable simplification, and they are mostly evaluative. While ANN is trained on the basis of real or numerical experiment data [4].

Due to the increasing complexity and heterogeneity of modern information and communication systems, traditional measures to ensure their functioning are increasingly untenable. One of the promising directions of solving practical problems in the field of information technology is the study of the possibilities of using ANN.

Analysis of the relevant literature showed that in the information environment neural networks have proven themselves in the following areas:

- network management and optimization;
- information security of communication networks;
- recognition of input information;
- information processing and search.

ANN successfully solves an important task in the field of telecommunications – finding the optimal path of traffic between nodes. Two features are taken into account: first, the solution must be adaptive, i.e. take into account the current state of the communication network and the presence of bad sections, and secondly, the optimal solution must be found in real time. In addition to flow routing control, neural networks are used to obtain effective solutions in the design of telecommunication networks [5], [6], [4].

However, the using of ANN technologies for solving practical problems is associated with many difficulties. One of the dominant problems in the application of ANNs models is the unknown architecture of the projected neural network and its degree of complexity, which will be sufficient for the reliability of the result [7–9].

2 Review of the Literature

In a number of works [10–26] was presented different algorithms to perform the ANNs training stage. The most common the Backpropagation method (BP), which allows you to adjust the weight of multi-layer complex ANNs using training sets. On the recommendation of E. Baum and D. Hassler [27, 28], the volume of the training set

is directly proportional to the number of all ANN weights and inversely proportional to the proportion of erroneous decisions in the operation of the trained network [13, 14].

It should be noted that the BP method was one of the first methods for ANNs training. Most of all brings trouble indefinitely long learning process. In complex tasks, it can take days or even weeks to train a network, and it may not train at all. The cause may be one of the following [10, 15, 16].

It should also be noted the possibility of retraining the network, which is rather the result of erroneous design of its topology. With too many neurons, the property of the network to generalize information is lost. The training set will be examined by the network, but any other sets, even very similar ones, may be misclassified.

The Backpropagation through time (BPTT) method has become a continuation, which is why it is faster. Moreover, it solves some of the problems of its predecessor. However, the BPTT experiences difficulties with local optima. In recurrent neural networks (RNN), the local optimum is a much more significant problem than in feed-forward neural networks. Recurrent connections in such ANNs tends to create chaotic reactions in the error surface, resulting in local optima appearing frequently. Also in the blocks of RNN, when the error value propagates back from the output, the error is trapped in the part of the block. This is referred to as the "error carousel", which constantly feeds the error back to each of the valves until they become trained to cut off this value. Thus, regular back propagation is effective when training an RNN unit to memorize values for very long durations [17, 18].

The main difference between genetic programming and genetic algorithms is that each individual in the population now encodes not the numerical characteristics that provide the optimality of the problem, but some solution to the problem. The term solution here refers to the configuration of the neural network.

Genetic algorithms have quite a significant list of advantages.

- Scalability. Genetic algorithms can be easily adapted for parallel and multicores programming, so that due to the peculiarities of this approach, the corresponding overhead costs are significantly reduced.
- Universality. Genetic algorithms do not require any information about the response surface, they work with almost any tasks.
- Genetic algorithms may be used for tasks in which the value of the fitness function changes over time or depends on various changing factors.
- Even in cases where existing techniques work well, interesting results can be achieved by combining them with genetic algorithms, using them as a complement to proven methods.
- Gaps existing on the response surface have little effect on the full efficiency of optimization, which also allows to further expand their use.

Thus, taking into account all the advantages and disadvantages of genetic algorithms, it is possible to obtain a sufficiently universal system for solving the necessary problems [29] and, in particular, for optimization of the neural network.

3 Sequential Modified Genetic Method of Recurrent Neural Networks Synthesis

In the method, which is proposed to find a solution using a population of neural networks: $P = \{NN_1, NN_2, ..., NN_n\}$, that is, each individual is a separate ANN $Ind_i \rightarrow NN_i$ [18–21]. During initialization population divided into two halves, the genes $g_{Ind_i} = \{g_1, g_2, ..., g_n\}$ of the first half of the individuals is randomly assigned $g_{Ind_i} = \{g_1 = \text{Rand}, g_2 = \text{Rand}, ..., g_n = \text{Rand}\}$. Genes of the second half of the population are defined as the inversion of genes of the first half $g_{Ind_i} = \overline{\{g_1 = \text{Rand}, g_2 = \text{Rand}, ..., g_n = \text{Rand}\}}$. This allows for a uniform distribution of single and zero bits in the population to minimize the probability of early convergence of the method ($p \rightarrow \min$) [30-34].

After initialization, all individuals have coded networks in their genes with-out hidden neurons (N_h), and all input neurons (N_i) are connected to each output neuron (N_o). That is, at first, all the presented ANNs differ only in the weights of the inter-neuron connection wi. In the process of evaluation, based on the genetic information of the individual under consideration, a neural network is first built, and then its performance is checked, which determines the fitness function ($f_{fitness}$) of the individual. After evaluation, all individuals are sorted in order of reduced fitness, and a more successful half of the sorted population is allowed to cross, with the best individual immediately moving to the next generation. In the process of reproduction, each individual is crossed with a randomly selected individual from among those selected for crossing. The resulting two descend-ants are added to the new generation $G = P` = \{Ind_1, Ind_2, ..., Ind_n\}$. Once a new generation is formed the mutation operator starts working. However, it is important to note that the selection of the truncation significantly reduces the diversity within the population, leading to an early convergence of the algorithm, so the probability of mutation is chosen to be rather large($p_{mut} = 15\text{-}25\%$) [35].

If the best individual in the population does not change for a certain number of generations (by default, it is proposed to set this number at eight), this individual is forcibly removed, and a new best individual is randomly selected from the queue. This makes it possible to realize the exit from the areas of local minima due to the relief of the objective function, as well as a large degree of convergence of individuals in one generation.

4 Parallel Genetic Modified Method for the Synthesis of Recurrent Neural Networks

Considering the features of the proposed modified genetic method for RNN synthesis, its parallel form can be represented as in Fig. 1. All stages of the method can be divided into 3 stages, separated by points of barrier synchronization. At the first stage,

the main core initializes the population *P*, and adjusts the initial parameters of the method, namely: the stopping criterion, the population size, the criterion for adaptive selection of mutations. Next, the distribution of equal parts of the population (subpopulations) and initial parameters to the cores of the computer system is performed. Initialization of the initial population cannot be carried out in parallel on the cores of the system, because the generated independent populations intersect thus increasing the search for solutions. The second stage of the proposed method is performed in parallel by the cores of the system. All cores perform the same sequence of operations on their initial population. After the barrier synchronization, the main core receives the best solutions from the other cores and checks the stopping criterion. If it is, then the next generation (*G*) is formed. Otherwise, after changing the initial parameters, allowing the cores of the system getting the other solutions, return to the distribution of the initial parameters to the cores on the system is performed. And then the cores perform parallel calculations according to the second stage of the method.

The proposed parallel method for RNN synthesis can be applied both on MIMD-systems [36] (clusters and supercomputers) and on SIMD (for example, graphics processors programmed with CUDA technology).

5 Experiments

The following hardware and software have been used for experimental verification of the proposed parallel genetic method for RNN synthesis [37]:

1. cluster of Pukhov Institute for Modeling in Energy Engineering National Academy of Sciences of Ukraine (IPME), Kyiv: processors Intel Xeon 5405, RAM – 4×2 GB DDR-2 for each node, communication environment InfiniBand 20Gb/s, middleware Torque and OMPI. MPI and Java threads programming models;
2. the computing system of the Department of software tools of Zaporizhzhya national technical university (ZNTU), Zaporizhzhya: Xeon processor E5-2660 v4 (14 cores), RAM 4x16 GB DDR4, the programming model of Java threads.
3. Nvidia GTX 960 graphics processor (GPU) with 1024 cores, which are programmed using CUDA technology.

During testing, the main task is to track the speed of the proposed method, quality and stability. Since synthesized RNN can be further used as diagnostic models for medical diagnosis, testing should be carried out on the relevant test data.

Data for testing were taken from the open repository – UC Irvine Machine Learning Repository. Data sample was used: public botnet datasets [38], particularly for the IoT. This dataset addresses the lack of public botnet datasets, especially for the IoT. It suggests real traffic data, gathered from 9 commercial IoT devices authentically infected by Mirai and BASHLITE. Originally aimed at distinguishing between benign and Malicious traffic data by means of anomaly detection techniques. However, as the malicious data can be divided into 10 attacks carried by 2 botnets, the dataset can also be used for multi-class classification: 10 classes of attacks, plus 1 class of 'benign'. Table 1 shows the main characteristics of the data sample.

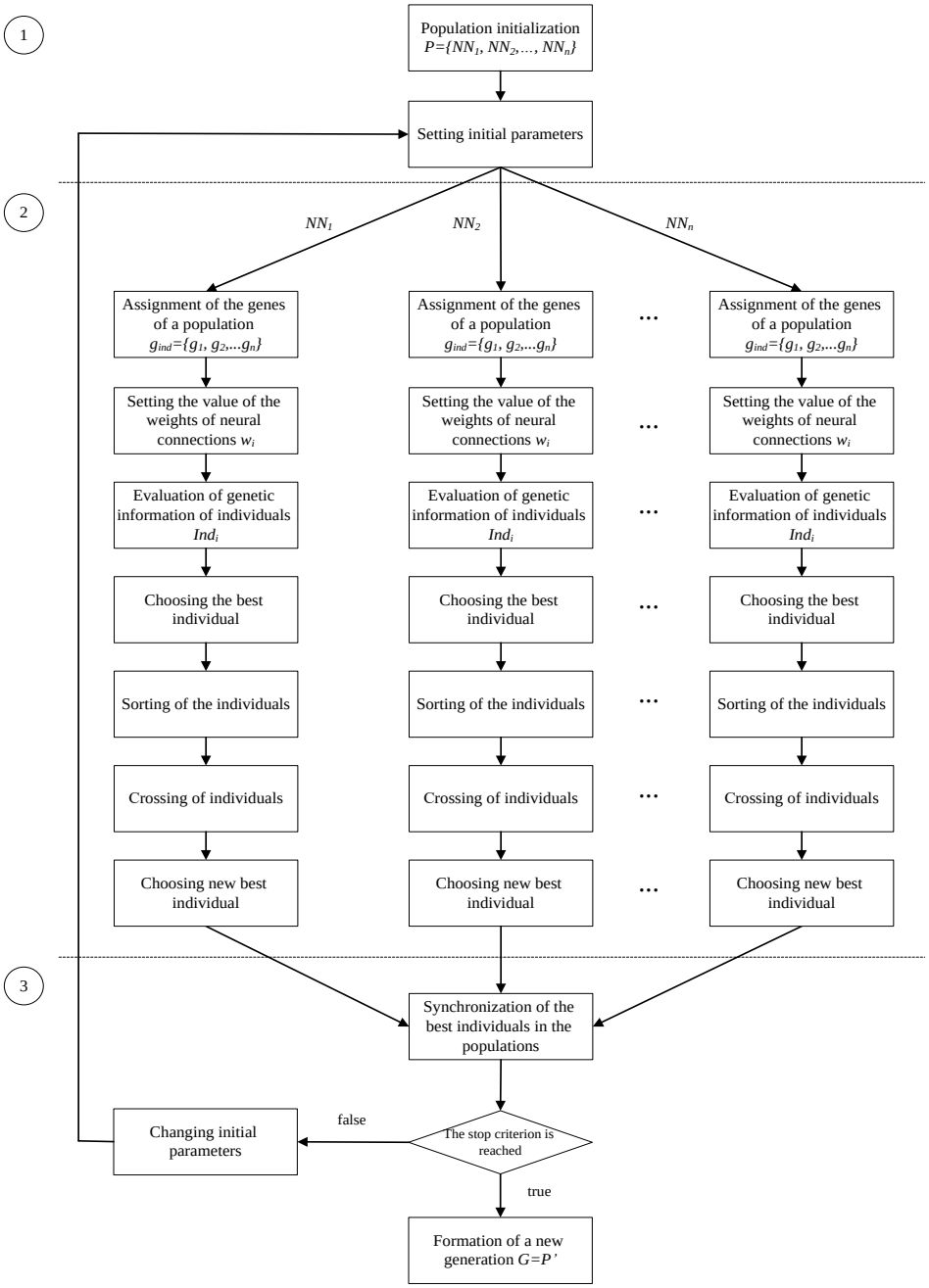

Fig. 1. Parallel genetic method for RNN synthesis

Table 1. Main characteristics of the datasets for IoT botnet attacks

Criterion	Characteristic	Criterion	Characteristic
Data Set Characteristics	Multivariate, Sequential	Number of Instances	7062606
Attribute Characteristics	Real	Number of Attributes	115

6 The Results Analysis

In the Fig. 2 and 3 are graphs of the execution time (in minutes) of the proposed method on computer systems, which depends on the number of cores involved. It can be seen from the graphs that the proposed method has an acceptable degree of parallelism and is effectively performed on both MIMD and SIMD systems. This way, the IPME cluster was able to reduce the method execution time from 1565 minutes (on one core) to an acceptable 147 minutes on 16 cores. On the ZNTU the computing system, the method execution time was reduced from 1268 minutes on a single core to 110 minutes on 16 cores. The differences in the performance of the systems are due to their architectural features: in the cluster cores are connected by means of the InfiniBand communicator, and in the multi-core computer they are located on a single chip, which explains the smaller impact of overhead (transfers and synchronizations). In addition, the processor in multi-core computer supports Turbo Boost technology [39], making the time of the method execution on the single core much less than the execution time on the core of the cluster that does not support this technology.
On a GPU with 960 cores involved, the execution time was 326.4 minutes, which can be adequately compared with the four cores of an IPME cluster or a ZNTU computing system.

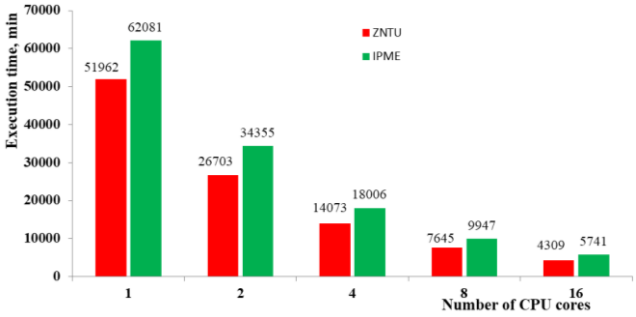

Fig. 2. Dependence the execution time of the proposed method to the number of involved cores of IPME cluster and ZNTU the computing system

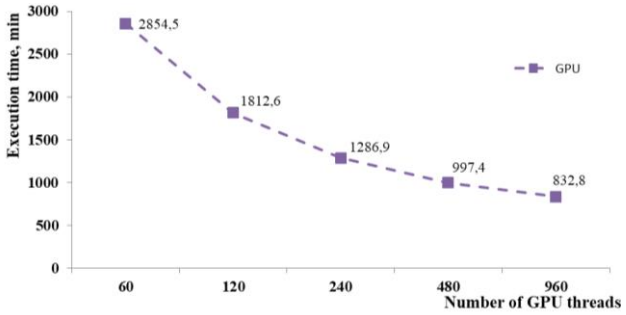

Fig. 3. Dependence the execution time of the proposed method to the number of GPU cores involved

The speedup graphics of calculations on a cluster IPME, ZNTU computing system and the GPU are shown in Fig. 4 and 5.

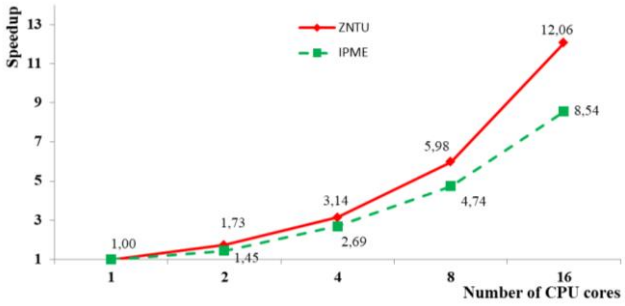

Fig. 4. The speedup graphics of calculations on a cluster IPME and ZNTU computing system

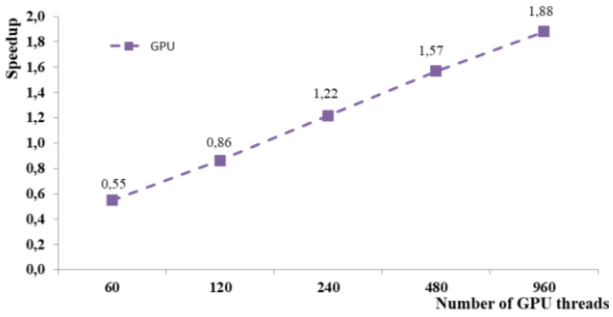

Fig. 5. The speedup graphics of calculations on a GPU

From the figures it is noticeable that the acceleration, though not linear, but approaches to linear. This is explained by the fact that communication overhead of the proposed method execution on computer systems is relatively small (Fig. 6, 7), and the number of parallel operations significantly exceeds the number of serial operations

and synchronizations. In communication overhead, is understood the ratio of the time spent by the system for transfers and synchronization among cores to the time of target calculations on a given number of cores.

The graph of efficiency of computer systems IPME and ZNTU is presented in Fig. 8. It shows that the using of even 16 cores of computer systems for the implementation of the proposed method retains the efficiency at a relatively acceptable level and indicates the potential, if necessary and possibly, to use even more cores.

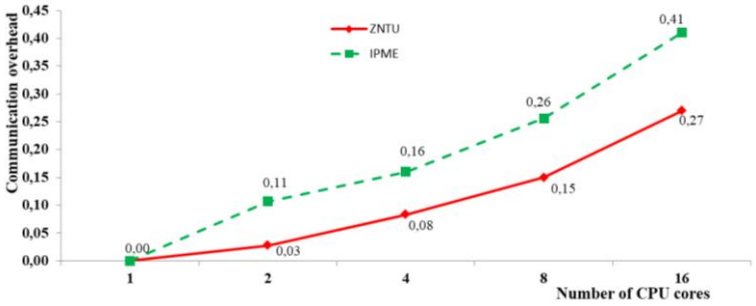

Fig. 6. Communication overhead performing the proposed method to the number of cores involved of IPME cluster and ZNTU the computing system

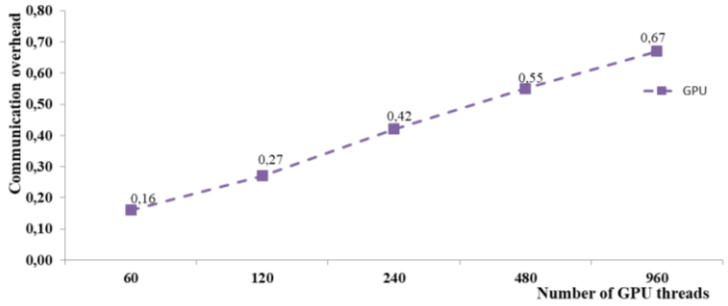

Fig. 7. Communication overhead performing the proposed method to the number of GPU cores involved

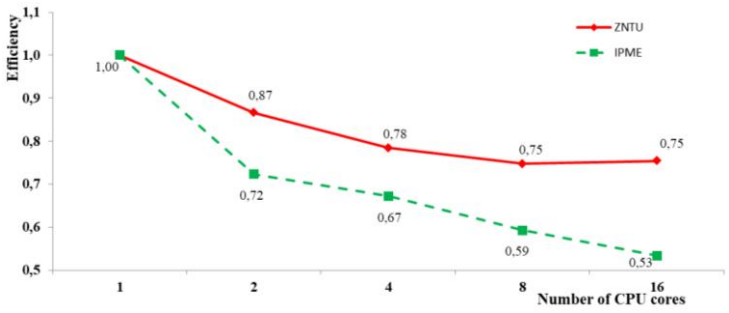

Fig. 8. The efficiency graph of IPME and ZNTU computing systems when executing the proposed method

Thus, the proposed method is well parallelized on modern computer architectures, which can significantly reduce the task: generate the models for future medical diagnosis execution time.

7 Conclusion

The problem of finding the optimal method of synthesis of ANN requires a comprehensive approach [40-43]. Existing methods of ANNs training are well tested [10-26], but they have a number of nuances and disadvantages. The paper proposes a mechanism for the use a modified genetic algorithm for its subsequent application in the synthesis of ANNs [44-51].

A model of parallel genetic method of RNS synthesis is proposed, which in comparison with the sequential implementation significantly speed up the synthesis process. In the developed model is proposed to parallelize the most resource-intensive operations: the generation of RNS populations, the calculation of genetic information about individuals, which can significantly accelerate the process of finding the best solution in the synthesis of networks.

Based on the analysis of the experimental results, it can be argued about the good work of the proposed method. However, to reduce iterativity and improve accuracy, it should be continued to work towards parallelization of calculations.

Acknowledgment

The work was performed as part of the project “Methods and means of decision-making for data processing in intellectual recognition systems” (number of state registration 0117U003920) of Zaporizhzhia National Technical University.

References

1. Ziaie, P.: Challenges and issues of ICT industry in developing countries based on a case study of the barriers and the potential solutions for ICT deployment in Iran. In: International Conference on Computer Applications Technology (ICCAT 2013), 1-6 (2013)
2. Aziz S., Dowling M. Machine Learning and AI for Risk Management. In: Lynn T., Mooney J., Rosati P., Cummins M. (eds) Disrupting Finance. Palgrave Studies in Digital Business & Enabling Technologies. Palgrave Pivot, Cham (2019)
3. Pukala R.: Use of Neural Networks in Risk Assessment and Optimization of Insurance Cover in Innovative Enterprises. Ekonomia i Zarzadzanie, 43-56 (2016). doi: 8. 10.1515/emj-2016-0023
4. Janowski, A., Nierebiński, P., Przyborski, M., Szulwic, J.: Object detection from lidar data on the urban area, on the basis of image analysis methods. Experiment on real data. In: 1st International Conference on Innovative Research and Maritime Applications of Space Technology, Gdansk, Poland (2015)
5. Nel, E. Omlin, C.: Machine Learning Algorithms for Packet Routing in Telecommunication Networks, Bellville, South Africa (2019).

6. Aicardi M., Davoli F., Minciardi R., Zoppoli R. The Use of Neural Networks in the Solution of Dynamic Routing Problems. In: New Trends in Systems Theory. Progress in Systems and Control Theory, vol 7 (1991)
7. Shkarupylo, V., Skrupsky, S., Oliinyk, A., Kolpakova T.: Development of stratified approach to software defined networks simulation. EasternEuropean Journal of Enterprise Technologies, vol. 89, issue 5/9, 67-73 (2017). doi: 10.15587/1729-4061.2017.110142
8. Leoshchenko, S., Oliinyk, A., Subbotin, S., Gorobii, N., Zaiko, T.: Synthesis of artificial neural networks using a modified genetic algorithm. Proceedings of the 1st International Workshop on Informatics & Data-Driven Medicine (IDDM 2018), 1-13 (2018). dblp key: conf/iddm/PerovaBSKR18
9. Kotsur, M., Yarymbash, D., Kotsur, I., Bezverkhnia, Yu.: Speed Synchronization Methods of the Energy-Efficient Electric Drive System for Induction Motors. IEEE: 14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET) 2018, 304-307, Lviv-Slavske, Ukraine (2018). doi:10.1109/TCSET.2018.8336208
10. Van Tuc, N.: Approximation contexts in addressing graph data structures. University of Wollongong Thesis Collection, 30-55 (2015)
11. Barkoulas, J. T., Baum, Ch. F.: Long Term Dependence in Stock Returns. EconomicsLetters, vol. 53, no. 3, 253-259 (1996)
12. Barkoulas, J. T., Baum, Ch. F., Travlos, N.: Long Memory in the Greek StockMarket. Applied Financial Economics, vol. 10, no. 2, 177-184 (2000)
13. Kolpakova, T., Oliinyk, A., Lovkin, V.: Improved method of group decision making in expert systems based on competitive agents selection. IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON), Institute of Electrical and Electronics Engineers, 939-943, Kyiv (2017). doi: 10.1109/UKRCON.2017.8100388
14. Stepanenko, O., Oliinyk, A., Deineha, L., Zaiko, T.: Development of the method for decomposition of superpositions of unknown pulsed signals using the second-order adaptive spectral analysis. EasternEuropean Journal of Enterprise Technologies, vol. 2, no 9, 48-54 (2018). doi: 10.15587/1729-4061.2018.126578
15. Handa, A., Patraucean, V.: Backpropagation in convolutional LSTMS. University of Cambridge Cambridge, 1-5 (2015)
16. Boden M.: A guide to recurrent neural networks and backpropagation. Halmstad University, 1-10 (2001)
17. Guo, J.: BackPropagation Through Time, 1-6 (2013)
18. Yue, B., Fu, J., Liang, J.: Residual Recurrent Neural Networks for Learning Sequential Representations. Information, 9, 56 (2018)
19. Erofeeva, V.A.: Review of the theory of data mining based on neural networks [Obzor teorii intellektualnogo analiza dannyih na baze neyronnyih setey. Stohasticheskaya optimizatsiya v informatike], 11 (3), 3-17 (2015)
20. Yarymbash, D., Kotsur, M., Subbotin, S., Oliinyk, A.: A New Simulation Approach of the Electromagnetic Fields in Electrical Machines. IEEE: The International Conference on Information and Digital Technologies, July 5th - 7th, Zilina, Slovakia, 2017, Catalog Number CFP17CDT-USB, 452-457, (2017). doi: 10.1109/DT.2017.8024332
21. Kowalski, P., Ukasik, S.: Training neural networks with krill herd algorithm. Neural Processing Letters, 1-13 (2015)
22. Mirjalili, S., Mirjalili, S., Hatamlou, A.: Multi-verse optimizer: a nature-inspired algorithm for global optimization. Neural Computing and Applications,1-19 (2015).
23. Cao, W., Wang, X., Ming, Z., Gao, J.: A review on neural networks with random weights. Neurocomputing, 275, 278-287 (2018)

24. Kieffer, B., Babaie, M., Kalra, S., Tizhoosh, H.R.: Convolutional neural networks for histopathology image classification: Training vs. using pre-trained networks. In 2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA), 1-6 (2017)
25. Sun, Z., Li, F., Huang, H.: Large scale image classification based on CNN and parallel SVM. International conference on neural information processing (2017)
26. Subbotin, S., Oliinyk, A., Skrupsky, S.: Individual prediction of the hypertensive patient condition based on computational intelligence. International Conference on Information and Digital Technologies, IDT 2015, 25 August 2015, Pages 348-356 (2015). doi: 10.1109/DT.2015.7222996
27. Yadav, J., Rani, A., Singh, V., Murari, B.: Prospects and limitations of non-invasive blood glucose monitoring using near-infrared spectroscopy. Biomedical Signal Processing and Control, Vol. 18 (2015). doi: 10.1016/j.bspc.2015.01.005
28. Smith-Miles, K. Gupta, J.: Neural Networks in Business: Techniques and Applications for the Operations Researcher. Computers and Operations Research (2000). doi: 27. 10.1016/S0305-0548(99)00141-0
29. Mitchell M. An introduction to Genetic Algorithm. MIT Press (1996)
30. Silver, D.: Mastering the game of Go without human knowledge. Nature 550, 354-359 (2017)
31. Bansal, T., Pachocki, J., Sidor, S., Sutskever, I., Mordatch, I.: Emergent complexity via multi-agent competition. In Proc. 2018 International Conference on Learning Representations (2018)
32. Miconi, T., Clune, J., Stanley, K.O.: Differentiable plasticity: training plastic neural networks with backpropagation. Proc. International Conference on Machine Learning, 3556-3565 (2018)
33. Liang, J., Meyerson, E., Miikkulainen, R.: Evolutionary architecture search for deep multitask networks. In Proc. Genetic and Evolutionary Computation Conference (GECCO), 466-473 (2018)
34. Zoph, B., Vasudevan, V., Shlens, J., Le, Q. V.: Learning transferable architectures for scalable image recognition. In Proc. IEEE Conference on Computer Vision and Pattern Recognition, 8697-8710 (2018)
35. Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. Neural Computation, vol. 9. issue 8, 1735-1780 (1997)
36. Skillicorn, D.: Taxonomy for computer architectures. Computer (21), 46-57 (1988). doi: 10.1109/2.86786
37. Alsayaydeh, J.A., Shkarupylo, V., Hamid, M.S., Skrupsky, S., Oliinyk, A.: Stratified Model of the Internet of Things Infrastructure, Journal of Engineering and Applied Science, vol. 13, issue 20, 8634-8638, (2018). doi: 10.3923/jeasci.2018.8634.8638
38. Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Breitenbacher, D., Shabtai, A., Elovici, Y.: N-BaIoT: Network-based Detection of IoT Botnet Attacks Using Deep Autoencoders, IEEE Pervasive Computing, Special Issue - Securing the IoT (2018)
39. Intel Turbo Boost Technology 2.0, https://www.intel.com/content/www/us/en/architecture-and-technology/turbo-boost/turbo-boost-technology.html
40. Oliinyk, A., Subbotin, S., Lovkin, V., Leoshchenko, S., Zaiko, T.: Development of the indicator set of the features informativeness estimation for recognition and diagnostic model synthesis. 14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET 2018), 903-908, (2018). doi: 10.1109/TCSET.2018.8336342

41. Oliinyk, A., Subbotin, S., Lovkin, V., Leoshchenko, S., Zaiko, T.: Feature Selection Based on Parallel Stochastic Computing," 2018 IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT), 347-351. Lviv (2018). doi: 10.1109/STC-CSIT.2018.8526729
42. Oliinyk, A., Leoshchenko, S., Lovkin, V., Subbotin, S., Zaiko, T.: Parallel data reduction method for complex technical objects and processes. 9th International Conference on Dependable Systems, Services and Technologies (DESSERT'2018), 526-532 (2018). doi: 10.1109/DESSERT.2018.8409184 IEEE Catalog number: CFP18P47-ART 978-1-5386-5903-8
43. Leoshchenko, S., Oliinyk, A., Subbotin, S., Zaiko, T.: Methods of semantic proximity extraction between the lexical units in infocommunication systems. 4th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T 2017), 7-13 (2017). doi: 10.1109/INFOCOMMST.2017.8246137
44. Callan, R.: The Essence of Neural Networks (The Essence of Computing Series). Prentice Hall, Europe (1999)
45. Gruau, F.: Genetic synthesis of Boolean neural networks with a cell rewriting developmental process. In Proceedings of the International Workshop on Combination of Genetic Algorithms and Neural Networks (COGANN-92). Los Alamos, CA: IEEE Computer Society Press, 55-74 (1992)
46. Moriarty, D., David, R., Miikkulainen, R.: Hierarchical evolution of neural networks. Evolutionary Computation Proceedings, 428-433 (1998). doi: 10.1109/ICEC.1998.699793.
47. Greer, B., Hakonen, H., Lahdelma, R., Miikkulainen, R.: Numerical optimization with neuroevolution. Evolutionary Computation CEC '02(1), 396-401 (2002). doi: 10.1109/CEC.2002.1006267
48. Enforced Subpopulations (ESP) neuroevolution algorithm for balancing inverted double pendulum, http://blog.otoro.net/2015/03/10/esp-algorithm-for-double-pendulum.
49. Stanley, K.O., Miikkulainen, R.: Evolving Neural Networks through Augmenting Topologies. The MIT Press Journals, vol. 10, num. 2, 99-127 (2002)
50. Whiteson, S., Whiteson, D.: Stochastic optimization for collision selection in high energy physics. Proceedings of the 19th national conference on Innovative applications of artificial intelligence, IAAI'07, vol. 2, 1819-1825 (2007)
51. Schmidhuber, J., Wierstra, D., Gagliolo, M., Gomez, F.: Training Recurrent Networks by Evolino. Neural computation. . Neural Computation, 19(3), 757-779 (2007)