Synthesis of Artificial Neural Networks Using a Modified Genetic Algorithm

Serhii Leoshchenko[1][0000-0001-5099-5518], Andrii Oliinyk[2][0000-0002-6740-6078],
Sergey Subbotin[3][0000-0001-5814-8268], Nataliia Gorobii[4][0000-0003-2505-928X]
and Tetiana Zaiko[4][0000-0003-1800-8388]

[1,2,3,4,5] Zaporizhzhia National Technical University, Dept. of Software Tools,
69063 Zaporizhzhia, Ukraine
[1]sedrikleo@gmail.com, [2]olejnikaa@gmail.com, [3]subbotin@zntu.edu.ua
[4]gorobiy.natalya@gmail.com, [5]nika270202@gmail.com

Abstract. This paper is devoted to the complex problem of synthesis of artificial neural networks. Firstly, the existing methods, recommendations and solutions of the problem are consider. As a new solution, the mechanism of using the modification of the genetic algorithm to determine the weights of the hidden and output layers (network training) is proposed. Testing and comparison of the results with the results of the existing methods were carried out for the correct evaluation of the method.

Keywords: artificial neural networks, synthesis, network training, recurrent connections, genetic algorithm, mesothelioma.

1 Introduction

In modern medicine, the main task for the use of information technology is to significantly improve the quality indicators in the diagnosis and therapy of various diseases. The existing methods and algorithms of modeling nonlinear systems are faced with problems of high dimensionality of tasks, the requirements of high accuracy and generalizing ability of the obtained models. These problems can be solved with the help of per-boron and iterative methods, which are based on the principles of selection, evolution and adaptation, which are methods of heuristic self-organization. At the same time, in real life it is quite difficult to create an adequate model of a complex object using only one method of inductive modeling. Usually it is required to combine modern methods and technologies of heuristic self-organization, to apply multilevel modeling, to develop hybrid algorithms.

Insufficiency or overabundance of data are frequent problems in solving tasks: there is not enough experiments (in the modeling of physical objects), not enough or difficult to allocate informative data on patients (to build a prediction of health). It is necessary to determine the parameters of the new element, to predict the outcome of the disease, to recommend treatment. Often artificial neural networks (ANNs) are used to solve such problems.

In a number of works [1-3] is noted are successfully applied of ANNs in various areas, including medicine, where the solution of problems of prediction, classification and management is required. It can be explained in way that ANNs have the possibility of nonlinear modeling in combination with a relatively simple implementation and this makes them indispensable in solving complex multidimensional problems.

However, despite all the advantages of ANN, there is a large number of difficulties in their implementation in medicine. Creation of ANNs is reduced to performance of the main steps:

- the choice of the structure of ANN;
- set weights of all neurons of the ANN (training of ANN) [4,5].

At the present to implement these tasks are not the rigorous methods of solution, there are only common recommendations. The proposed methods are aimed at solving local problems, which often leads to an unsatisfactory structure of ANNs and a significant training time.

In this paper, the authors propose a method of using a genetic algorithm for the synthesis of ANNs – creation of the structure and configuration of the network weights (learning with the teacher).

2 Review of the literature

The basis of ANNs are neurons with a structure similar to biological analogues. Each neuron can be represented as a microprocessor with several inputs and one output. When neurons are joined together, a structure is formed, which calls a neural network. Vertically aligned neurons form layers: input, hidden and output. The number of layers determines the complexity and, at the same time, the functionality of the network, which is not fully investigated.

For researchers, the first stage of creating a network is the most difficult task. The following recommendations are given in the literature [4–6].

1. The number of neurons in the hidden layer is determined empirically, but in most cases the rule is used $N_h \le N_i + N_o$, where N_h is the number of neurons in the hidden layer, N_i in the input and N_o output layers.
2. Increasing the number of inputs and outputs of the network leads to the need to increase the number of neurons in the hidden layer.
3. For the ANNs modeling multistage processes required additional hidden layer, but, on the other hand, the addition of hidden layers may lead to overwriting and the wrong decision at the output of the network.

Based on these recommendations, the number of layers and the number of neurons in the hidden layers is chosen by the researcher, based on his personal experience.

In a number of works [6–16] 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 [7, 8], 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 [9, 10].

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 [6, 11, 12].

1. Network paralysis. During network training, the weights can become very large as a result of the correction. This can cause all or most neurons to function at very high OUT values, in an area where the derivative of the compression function is very small. Since the error sent back in the learning process is proportional to this derivative, the learning process can practically freeze.
2. Local minimum. The network can hit a local when there are much deeper lows nearby. At the point of the local minimum, all directions lead up, and the network is unable to get out of it. Statistical training techniques can help avoid this trap, but they are slow.
3. Step size. The step size should be taken as final. If the step size is fixed and very small, the convergence is too slow, if it is fixed and too large, paralysis or constant instability may occur.

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 [13, 14].

The Hebb method does not guarantee the convergence of the learning process, i.e. the error of approximation of the function by the neural network may exceed the permissible value. The main disadvantage of the using Hebb method is that the convergence of the algorithm decreases with increasing dimension n of the input vector. For $n > 5$, it is difficult to guarantee convergence. This method is usually used as some element in other learning algorithms. The most preferable is the use of the Hebb rule in unsupervised learning algorithms [6, 15].

For the training by connectionist temporal classification (CTC) [16] it should be noted the main problem of reinforcement learning: random re-training to do in such conditions one and the same action. Sometimes it is possible to mistakenly associate

the reaction of the environment with the action that immediately preceded this reaction. This effect was later called “superstition” in the pigeon [17]. The above problem is the so-called dilemma of “exploitation vs exploration”, that is, on the one hand, you need to explore new opportunities, to study the environment so that it is something interesting to find. On the other hand, at some point you can decide that everything is investigated and move on – there is no need.

The above problems explain the urgency of developing a new approach to the synthesis of ANNs. The main purpose of the authors is to research and test a new method to solve the second problem, namely, training ANNs, which will be based on the use of genetic algorithms, as a means of determining the weights of the hidden and output layers.

3 Materials and methods

In the method, which is proposed to find a solution using a population of neural networks, that is, each individual is a separate ANN [18–20]. During population initialization, one half of the individuals is randomly assigned. Genes of the second half of the population are defined as the inversion of genes of the first half of individuals. This allows the “1” and “0” bits to be evenly distributed in the population to minimize the likelihood of early convergence of the algorithm.

After initial initialization, all individuals have coded networks in their genes without hidden neurons, and all input neurons are connected to each output neuron. That is, at first, all the presented ANNs differ only in the weights of the interneuron bonds. 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 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 descendants are added to the new generation. 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, about 15-25% [21].

If the best individual in the population does not change for more than 7 generations, the algorithm is restarted. During the restart, the entire population is reinitialized and the solution search process starts from scratch. 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.

3.1 Using of genetic operators

It is obvious that the chosen method requires special genetic operators that implement crossover and mutation.

At crossover two parental individuals which produce two descendants are used. Common neurons and connections are inherited by both offspring, and the value of connections in the networks of descendants are formed by a two-point crossover. Elements of ANN, of distinct "played out" between generations.

An important feature is that neurons with the same indices are considered identical, despite the different number of connections and position in the network, as well as the fact that one of these neurons could have a different index, which changed as a result of correction of indices after mutation. For this purpose, two coefficients were introduced that regulate the size and direction of the network.

The first of them characterizes the degree of "connectedness" of neurons in the network and is calculated by the formula:

$$f_c = \frac{N_c}{2^{FB-1}\left[N_s\left(N_s-1\right)-N_i\left(N_i-1\right)-\left(1-FB\right)N_o\left(N_o-1\right)\right]}, \quad (1)$$

where N_c is the number of connections in the network, N_i, N_o, N_s are respectively, the number of input, output neurons and the total number of neurons in the network, FB is a variable indicating the permitted occurrence of feedbacks (FB =1) or not (FB =0). It is worth noting that connections from hidden neurons to the output can appear in any case. Thus, the smaller f_c the more likely a new relationship will be added as a result of the mutation. There are 4 possible uses f_c:

- calculated by the formula;
- squared;
- multiplied by some factor;
- squared and multiplied by some factor.

The use of the second coefficient is based on the assumption that the more elements in the sum of the input and output vectors of the training choice (the greater the total number of input and output neurons), which is probably a more complex network is necessary to solve the problem. The second coefficient is calculated by the formula:

$$f_n = \frac{N_i + N_o}{N_s}. \quad (2)$$

That is, the more neurons in the network, the less will be f_n and the less likely will be selected mutation that adds a new hidden neuron. As well as f_c there are possible 4 use cases f_n:

- calculated by the formula;

- squared;
- multiplied by some factor;
- squared and multiplied by some factor.

For any of the described cases, the algorithm uses a ligament $f_n \cdot f_c$, because for use it is necessary to take into account the degree of connectivity of already existing neurons.

Thus, using mutations can be "pointwise" to change the parameters of the structure of the ins.

Chaotic the addition (removal) of neurons and connections can lead to situations where, for example, in a network of many neurons and few connections. It would be more logical to apply different types of mutations depending on the features of the network architecture represented by the mutating individual.

Removing links ins gives a side effect: there may be "hanging" neurons that have no incoming connections, as well as dead-end neurons, that is, without output connections. In cases where the function of neuronal activation is such that at zero weighted sum of inputs its value is not equal to zero, the presence of "hanging" neurons makes it possible to adjust the neural displacement. It is worth noting that, on the other hand, the removal of links may contribute to the removal of some uninformative and uninformative input features.

3.2 Choosing the mutation type

Consider the dependence of the type of mutation on the values f_c and f_n. Adaptive mutation mechanism is one of the key features of the proposed method.

The choice of mutation type is determined based on the values of f_c and $f_n \cdot f_c$. This approach, on the one hand, does not limit the number of hidden neurons "from above", on the other hand, it prevents the immeasurable increase of the network, because the addition of each new neuron to the network will be less likely. The mutation of the weight of a random existing bond occurs for all mutating individuals with a probability of 0.5.

Let us consider in more detail how to choose the type of mutation. Fig. 1 shows the block diagram of the selection of the type of mutation. Here RV is a random variable, N_h is the number of hidden neurons in the mutating network. For short the selection and calculation f_c and f_n, and mutation of the weight by accident you select the relationship in the schema is not specified.

Conventionally, the entire algorithm can be divided into two "branches" on the first conditional transition:

1. branch increase f_c is carried out for the fulfilment of the conditions of transition;
2. branch reduction f_c, performed if the transition condition is not met.

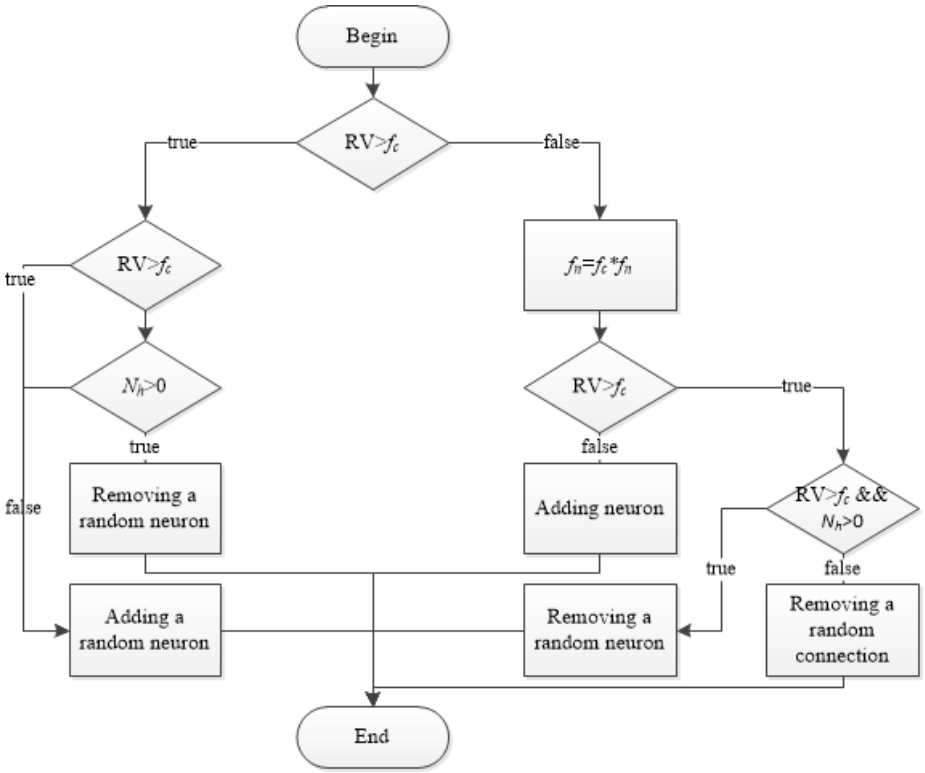

Fig. 1. The diagram of the selection of the type of mutation

Since the removal of a neuron can lead to both reduction and increase depending on the number of connections of the neuron, this option is present in both branches. Thus, the main factor for the regulation of the structure of the obtained ANN is the degree of their "connectivity". In other words, how fully the possible network connections are implemented.

Multiplication f_n by f_c is necessary in order to change the number of neurons adequately network topology, because the addition (removal) of neurons need information about the feasibility of changes. This information can be obtained indirectly from the value of the characteristic.

3.3 The calculation of the output layer of ANN

The value of the mean square error is replaced by the criterion of maximum separation of support vectors. In this case, the optimal linear weights can be estimated using quadratic programming, as in the traditional support vector machine.

One of the problems of neuroevolutionary method realization is the algorithm of ANN output calculation with arbitrary topology.

ANN can be represented as a directed planar graph. Based on the fact that the network structure can be any, loops and cycles containing any nodes are allowed in the graph, except for the nodes of the corresponding input neurons. Let denote the set of nodes of the graph by $V = \{v_i \mid i \in [0; N_v - 1]\}$, and a set of arcs through $E = \{e_j \mid j \in [0; N_e - 1]\}$, where N_v and N_e are accordingly, the number of nodes and arcs in the graph, and $N_v \equiv N_s$, and $N_e \equiv N_c$. The arc, which goes from node *k* to node 1 denote by an ordered pair $e_{k,l} = (v_k, v_l)$, the weight of the corresponding link will be denoted by $w_{k,l}$.

Give the index to the nodes of the graph as neurons, that is, the nodes that are the input neurons, called input. have an index out of range $[0; N_l - 1]$. By analogy, the indexes of outgoing nodes belong to the interval $[N_l; N_l + N_o - 1]$, and indexes for hidden nodes will be set in the interval $[N_l + N_o; N_v - 1]$.

Let introduce an additional characteristic for all nodes of the graph equal to the minimum length of the chain to any of the input nodes and denote it l_i. Let's call l_i the layer to which the i^{th} node belongs. Thus, all input nodes belong to the 0^{th} layer, not all input nodes that have input arcs from the input belong to the 1st layer, all other nodes with input arcs from nodes of the 1st layer will belong to the layer with index 2, etc .in this case, there may be situations when the node does not have input arcs, we will call it a hanging node with the layer number $l_i = -1$.

For arcs, we also introduce an additional characteristic $b_{k,l}$ for the arc $e_{k,l}$, which is necessary to determine whether the arc corresponds to forward or reverse. It will be calculated as follows:

$$b_{k,l} = \begin{cases} 1, l_l - l_k > 0 \\ -1, l_l - l_k \le 0 \end{cases}. \tag{3}$$

That is, if the index of the layer of the end node of the arc is greater than the index of the layer of the beginning node, then we will consider such an arc as a straight line, otherwise we will consider the arc as an inverse.

Since each node of the graph represents a neuron, we denote by sum_i the value of the weighted sum of inputs, and through o_i is the value of the output (the value of the activation function of the i^{th} neuron-node). Then, $o_i = f(sum_i)$ where *f* is the function of neuron activation.

Let's divide the whole process of signal propagation from the input nodes into stages, and during one such stage the signals “manage” to pass only one arc. The number of the stage is denoted by *s*. For the very first stage *s=1*. For short assumed that all arcs have the same length, and the signals are sewn on them instantly. We denote the feature that the output of node *i* was updated at this stage through a_i, that

is, if $a_i \equiv 1$, then the output of the node at stage *s* is calculated, otherwise, if $a_i \equiv 1$ – not.

Let's introduce one more designation $X = \{x_i \mid i \in [0; N_l - 1]\}$ it is vector of input signals. Then the algorithm for calculating the ANN output is as follows:

1. $o_i = x_i$, $a_i = 1$, for all $i \in [0; N_l - 1]$;
2. $o_i = 0$, for all $i \in [N_l; N_s - 1]$;
3. *s=1*;
4. $sum_i = 0$, $a_i = 1$, for all $i \in [N_l; N_s - 1]$;
5. if $s \equiv 1$, than go to the step number 7;
6. calcultion of the feedback network. For all input reverse arcs $e_{j,k}$ node v_k, where $k \in [N_l; N_s - 1]: sum_k = sum_k + o_j$, if $l_j < s$;
7. if $a_i \equiv 0$, than $fn(i)$ for all $i \in [N_l; N_s - 1]$;
8. if the stop criterion is not met, than *s=s+1* and go to the step number 4.

Here $fn(i)$ is a recursive function that calculates the output of the 1st node taking into account all straight arcs. Works on the following algorithm:

1. if $l_i < 0$, than go to the step nuber 3;
2. for all input arcs $e_{k,l}$ node v_i: if $a_k = 1$, than $sum_i = sum_i + o_k$, else $fn(k)$;
3. $o_i = f(sum_i)$;
4. exit.

The stopping criterion of the ANN output calculation algorithm can be one of the following:

- stabilization of values at the output of ANN;
- *s* exceeds the set value.

It is more reliable to calculate the output until the values at the output of ANN do not change, but for the case when the network contains cycles and/or loops, its output may never become stable. Therefore, the required additional stopping criteria limiting the maximum number of stages of calculation of network output. For networks with no feedback (FB =0) in many cases, allow the $\max(l_i)+1$ phases.

4 Experiments

For a full assessment, we will conduct a series of experiments. A sample of data on patients diagnosed with Mesothelioma will be used as data. The sample is publicly available [22] and provided by Dicle University Faculty of Medicine in Turkey. The main characteristics of the sample are given in Table 1.

Table 1. Characteristics of the sample data

Criterion	Characteristic	Criterion	Characteristic
Data Set Characteristics	Multivariate	Number of Instances	324
Attribute Characteristics	Real	Number of Attributes	34

For the correct evaluation of the experimental results, will be compared the developed method with the BP method and BPTT method. It should be taken into account that the inverse distribution of the error will be compared under the condition *FB*=0, and the distribution in time with the condition *FB*=1. This is due to the fact that each of the methods is used for the synthesis of different types of ANNs (with and without recurrent connections).

Mesothelioma is a rare, aggressive form of cancer that develops in the lining of the lungs, abdomen, or heart. Caused by asbestos, mesothelioma has no known cure and has a very poor prognosis [23].

5 The results analysis

During the experiments, the special attention was paid to the main characteristics of the methods, namely: the time spent, the boundary values of the error, the mean value of the error and the average error of final network. The results are shown in table 2.

Table 2. Experimental result

	Time	E_{min}	E_{max}	E	Average Error of Final Network
BP	468.013s	0.1699	1.9398	0.4639	0.4402
Modified GA (FB=0)	335.843s	0.3417	0.3846	0.35	0.2488
BPTT	9389.55s	0.2992	0.3212	0.3046	0.3067
Modified GA (FB=1)	631.373s	0.3058	0.3054	0.3102	0.2962

As experiments have shown, in all cases, the modified genetic method showed better results than existing methods. On the other hand, it should be noted that the difference in the mean error of the finite network in the second group of experiments (when comparing the back propagation in time and the modified genetic algorithm) is not so great. To improve the operation of the proposed method in the case of recurrent ins, it is possible to resort to the strategy of using support vector machine (SVM) to clarify the weights of the output layer, as proposed in [24, 25].

Fig. 2 shows the distribution of iterations during the experiments.

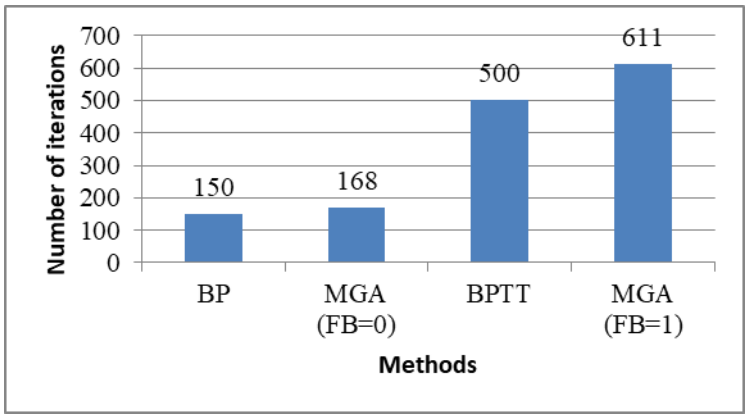

Fig. 2. The diagram of distribution of number of iterations

As can be seen from the diagram, the modified genetic method was more highly iterative than the existing methods, but the time spent on the iteration was less. That is, we can conclude that the iterations are not complex and to reduce them we can resort to parallelization [26–29].

6 Conclusion

The problem of finding the optimal method of synthesis of ANN requires a comprehensive approach. Existing methods of ANNs training are well tested, 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. 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 and the using of SVM.

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. Liu, P., Li, Y., El Basha, M.D., Fang, R.: Neural Network Evolution Using Expedited Genetic Algorithm for Medical Image Denoising. MICCAI 2018: Medical Image Computing and Computer Assisted Intervention, pp.12–20. Springer, Nature Switzerland (2018).

2. Zolin, A.G., Silaeva, A.Yu.: Primenenie neyronnyih setey v meditsine. Aktualnyie problemyi nauki, ekonomiki i obrazovaniya XXI veka, 264-271 (2012).
3. Churyumova, I.G.: Meditsinskaya sistema prinyatiya resheniy s ispolzovaniem neyronnoy seti, 199-202 (2004).
4. Bondarenko, I.B., Gatchin, Yu.A., Geranichev, V.N.: Cintez optimalnyih iskusstvennyih neyronnyih setey s pomoschyu modifitsirovannogo geneticheskogo algoritma. Nauchno-tehnicheskiy vestnik informatsionnyih tehnologiy, mehaniki i optiki, vol. 2 (78), 51–55 pp. (2012).
5. Lukichev, D.V., Usoltsev, A.A.: Sintez optimalnoy strukturyi neyrosetevyih ustroystv, 97–102 (2005).
6. Van Tuc, N.: Approximation contexts in addressing graph data structures. University of Wollongong Thesis Collection, 30–55 (2015).
7. Barkoulas, J. T., Baum, Ch. F.: Long Term Dependence in Stock Returns. EconomicsLetters, vol. 53, no. 3, 253–259 pp. (1996).
8. Barkoulas, J. T., Baum, Ch. F., Travlos, N.,: Long Memory in the Greek StockMarket. Applied Financial Economics, vol. 10, no. 2, 177–184 pp. (2000).
9. Shkarupylo, V., Skrupsky, S., Oliinyk, A., Kolpakova, T.: Development of stratified approach to software defined networks simulation. EasternEuropean Journal of Enterprise Technologies, vol. 5, no9, 67–73 pp. (2017). doi: 10.15587/1729-4061.2017.110142.
10. Stepanenko, A., 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 pp. (2018). doi: 10.15587/1729-4061.2018.126578.
11. Handa, A., Patraucean, V.: Backpropagation in convolutional LSTMS. University of Cambridge Cambridge 1–5 pp. (2015).
12. Boden M.: A guide to recurrent neural networks and backpropagation. Halmstad University 1–10 pp. (2001).
13. Guo, J.: BackPropagation Through Time 1–6 pp. (2013).
14. Yue, B., Fu, J., Liang, J.: Residual Recurrent Neural Networks for Learning Sequential Representations. Information, 9, 56 (2018).
15. Erofeeva, V.A.: Obzor teorii intellektualnogo analiza dannyih na baze neyronnyih setey. Stohasticheskaya optimizatsiya v informatike 11 (3), 3–17 pp. (2015).
16. Graves, A., Fernández, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In Proceedings of the 23rd international conference on Machine learning (ICML '06). ACM, New York, 369–376 pp. 2006). DOI: https://doi.org/10.1145/1143844.1143891.
17. Skinner, B.F.: "Superstition" in the pigeon. Journal of Experimental Psychology vol. 38(2), 168–172 pp. (1948). doi: 10.1037/0096-3445.121.3.273.
18. Chiroma, H., Mohd Noor, A.S., Abdulkareem, S., Abubakar, A.I., Hermawan, A., Qin, H,, Hamza, M,F., Herawan, T.: Neural Networks Optimization through Genetic Algorithm Searches: A Review. Applied Mathematics & Information Sciences, vol. 11 (6), 1543–1564 pp. (2017).
19. Tsoy Yu.R. Razrabotka geneticheskogo algoritma nastroyki iskusstvennoy neyronnoy seti. Tomskiy politehnicheskiy unIversitet (2004).
20. Using Genetic Algorithm for optimizing Recurrent Neural Network, http://aqibsaeed.github.io/2017-08-11-genetic-algorithm-for-optimizing-rnn/
21. 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 2017), 939–943 pp. (2017). doi: 10.1109/UKRCON.2017.8100388.
22. UCI Machine Learning Repository, Mesothelioma disease data set Data Set, https://archive.ics.uci.edu/ml/datasets/Mesothelioma%C3%A2%E2%82%AC%E2%84%A2s+disease+data+set+.
23. Mesothelioma Cancer, https://www.mesothelioma.com/mesothelioma/.
24. Schmidhuber, J., Wierstra, D., Gagliolo, M., Gomez, F.: Training Recurrent Networks by Evolino. Neural computation. vol. 19(3), 757–779 pp. (2007). doi: 10.1162/neco.2007.19.3.757.
25. 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 pp. (2018). doi: 10.1109/TCSET.2018.8336342.
26. Togelius, J., Schaul, T., Schmidhuber, J., Gomez, F.: Countering Poisonous Inputs with Memetic Neuroevolution. In: Günter Rudolph, Thomas Jansen, Simon Lucas, Carlo Poloni, and Nicola Beume (eds.) Proceedings of the 10th international conference on Parallel Problem Solving from Nature: PPSN X,. Springer–Verlag, Berlin, Heidelberg, 610–619 pp. (2008). doi=http://dx.doi.org/10.1007/978-3-540-87700-4_61.
27. 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 pp. (2017). doi: 10.1109/INFOCOMMST.2017.8246137.
28. Eiben, A. E., Raué, P.-E., Ruttkay Z.: Genetic algorithms with multi-parent recombination. In: Yuval Davidor, Hans-Paul Schwefel, and Reinhard Männer (Eds.) Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature: Parallel Problem Solving from Nature (PPSN III). Springer-Verlag, London, UK, UK, 78–87 pp. (1994).
29. 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 pp. (2018). doi: 10.1109/DESSERT.2018.8409184 IEEE Catalog number: CFP18P47-ART 978-1-5386-5903-8.