Parallel genetic method for the synthesis of recurrent neural networks for using in medicine

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 Viktor Lytvyn[5[0000-0003-4061-4755]]

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

Abstract. In modern medicine, information technologies are widely used in the diagnosis and treatment of various diseases. The main task of creating such systems is to improve the quality of diagnosis and treatment. Therefore, the work aimed at finding new solutions in the creation of such systems are relevant. Despite all the advantages of neural networks, there are many difficulties in their implementation in medicine. In this paper are presented methods for solving the problem of recurrent neural network synthesis, which can be used as models in medical diagnostics.

Keywords: medical diagnosis, neural networks, synthesis, parallel, genetic method.

1 Introduction

Intensive development of medical science, expansion of possibilities of deepening in etiology, pathogenesis of disease, increase in data on markers of various pathological conditions dictates the necessity ща searching new approaches to the processing of the results. Today, it is important to quickly analyze a large number of data and make the right decision, which can affect the prognosis, course and outcomes of the disease. In this case, more and more attention is paid to information technologies (IT), and in the scope of medicine it can be talked about electronic medicine [1]. IT is implemented in the form of special medical systems for various purposes and individual automated diagnostic and treatment devices.

The using IT allows to solve various tasks, which include prediction the risks of diseases, complications and treatment effectiveness, early diagnosis, treatment planning, monitoring the patient's health, automated analysis and statistical processing of clinical material. Medical systems significantly simplify the work in such situations when it is impossible to present the problem in numerical form, there is no certainty

or accuracy in the studied parameters or there is no one-digit algorithm for solving problems [2]. These characteristics are suitable for solving medical problems, which are a large amount of multidimensional, complex and sometimes contradictory clinical data obtained in the process of censored observations.

Currently, the use of statistical methods of data processing prevails in medical research. The most common descriptive methods used in traditional statistical studies are survival analysis and multivariate complex analysis classified as discriminant, cluster, factor, and correlation.

The fact that artificial neural network (ANN) are very successfully used in various fields, where it is necessary to solve the problems of forecasting, classification and management, explaine the undying interest to the ANN methods, which has been observed recently. ANN have the ability to nonlinear modeling in combination with a relatively simple implementation and this makes them indispensable in solving complex multidimensional problems, including medical [3].

Today, there are many models of using ANN's architectures, which differ in their computational complexity, the degree of similarity with the living neurons of the brain, as well as having an exclusive and unique in its creation. Therefore, ANN are not subject to any classification standards in comparison with traditional statistical methods.

Existing ANN are able to work with both numerical data lying in a certain limited range and non-numerical parameters, for example, graphic images of various configurations. However, the non-standard scale of quantitative characteristics, the presence of missing values, the variability of nominal variables, the conversion of qualitative parameters into a numerical function or declaring them insignificant, create additional problems in the operation of the ANN and distort the output result.

From both scientific and practical point of view, one of the main advantages of using ANN is its ability to learn with data analysis, the establishment of complex and hidden connections and the subsequent presentation of independent results [4], [5]. In the process of training, when a large number of errors appear, it is possible to revise both the configuration of the network itself and to change the parameters included in its training [1].

Thus, the advantages of using ANN are:

- ability to learn from multiple qualitative and quantitative examples with unknown patterns between input and output data without fragmentation of the data sample. A more accurate description of the parameters, the ability to display the dynamics of the statistical properties of various indicators;
- effective data compression due to the construction of nonlinear mappings and the ability to visualize in the space of a smaller number of nonlinear principal component neural networks built;
- ability to make decisions based on absolute resistance to noise of input data and adaptation to environmental changes;
- modelling real situations solving tasks is done by analyzing the knowledge from their own experience of the ANN for an independent but. Minimal or complete absence of subjective factor influence on the final result. The ability to manually edit

the values of individual parameters and their properties of an ANN, as well as other ways to include expert knowledge in the network;
- potential fault tolerance in hardware implementation of ANN;
- the possibility of using in situations that require the immediate adoption of the solution.

However, the use of ANN technologies for solving practical problems is associated with many difficulties. One of the dominant problems in the application of ANN's 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.

2 Review of the literature

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 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.

At the moment, there are several reasons to criticize the genetic algorithm and genetic programming using. Below is a list of the main drawbacks of this approach.

- Re-evaluation of fitness function for complex problems is often a factor limiting the use of artificial evolution algorithms. Finding the optimal solution for a complex high-dimensional problem often requires a very costly evaluation of the fitness function.
- Genetic algorithms are poorly scalable to the complexity of the problem to be solved.
- Despite attempts to formalize genetic algorithms and the neuroevolutionary approach in particular [17], the theoretical basis remains scant.
- The solution is only more suitable than other solutions. As a result, the stopping criterion of the algorithm is unclear for each problem.
- In many problems, genetic algorithms tend to converge to a local optimum or even to controversial points, instead of a global optimum for the given problem [18].

However, many of these shortcomings can be corrected. For example, to prevent premature convergence, it is necessary to correctly select such parameters of the genetic algorithm as the population size and the percentage of individuals subjected to mutation. In addition, new variants of genetic operators are constantly being developed. The additional cost of recalculating the fitness function value can be avoided by using flags for cases where the fitness function value does not change over time.

However, in addition to the above drawbacks, 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 [19] 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 $Ind_i \rightarrow NN_i$ [18–20]. 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 ($p \rightarrow \min$).

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 interneuron 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} = 15-25%) [20].

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. The general scheme of the method demonstrated at Fig.1.

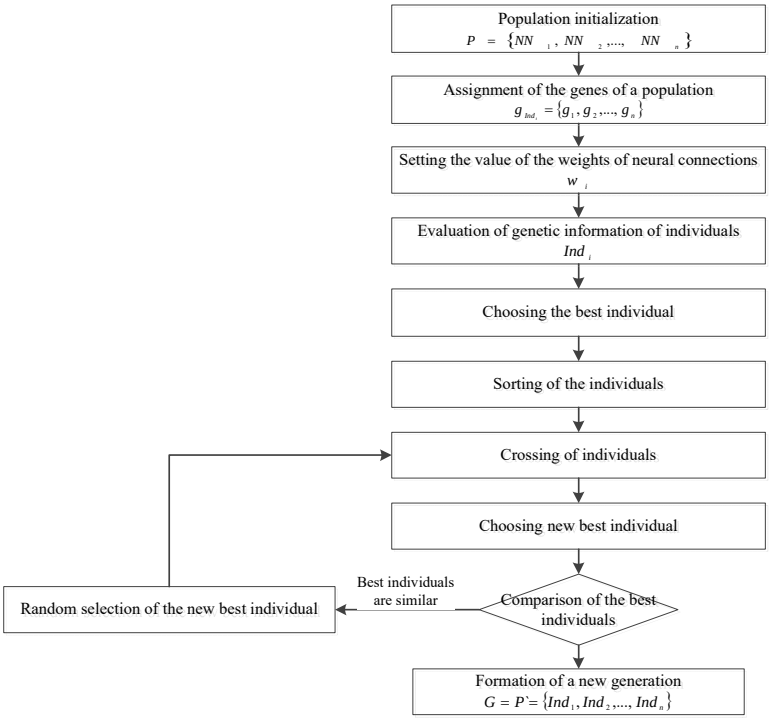

Fig. 1. The general scheme of the method

3.1 The calculation of the output layer of ANN

On condition using the support vector machine, the optimality criterion for calculating the output weights may not be specified. If the value of the mean square error is replaced by the criterion of the maximum separation of the support vectors, then the optimal linear weights of the output can be estimated using, for example, quadratic programming, as in the traditional method of support vectors, for this it is advisable to use the Evoke operator [21], by the formula:

$$y(t) = w_0 + \sum_{i=1}^{k} \sum_{j=0}^{l_i} w_{i,j} K\left(\phi(t), \phi^i(j)\right), \quad (1)$$

where $\phi(t) \in R^n$ is the output of a recurrent neural network $f(\cdot)$ at a time t; $K(\cdot,\cdot)$ is a predefined kernel function; $w_{i,j}$ is weights corresponding to k training sequences ϕ^i, each length l_i, and are calculated using the support vector machine.

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 $c_{k,l} = (v_k, v_l)$, 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 ch_i. Let's call ch_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 $ch_i = -1$.

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

$$b_{k,l} = \begin{cases} 1, ch_l - ch_k > 0 \\ -1, ch_l - ch_k \le 0 \end{cases} \quad (2)$$

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, $o_i = f_{fitness}(sum_i)$ where $f_{fitness}$ 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 $a_i \equiv 1$ is 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 feedbacks $c_{j,k}$ node v_k, where $k \in [N_l; N_s - 1]$: $sum_k = sum_k + o_j$, if $ch_j < s$;
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.

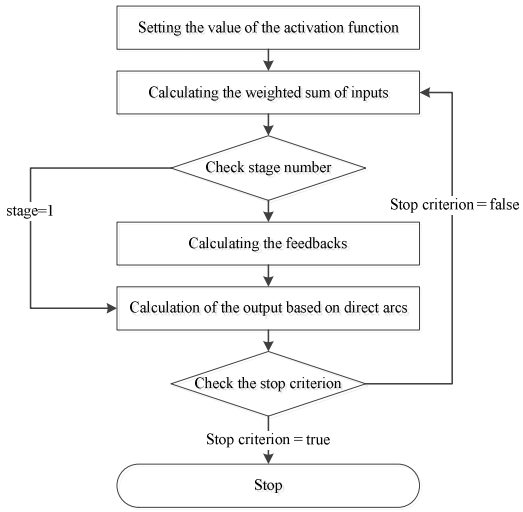

Fig. 2. The general scheme of the calculation of the output layer of ANN

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 $ch_i < 0$, than go to the step number 3;
2. for all input arcs $c_{k,l}$ 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.

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. 3. 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.

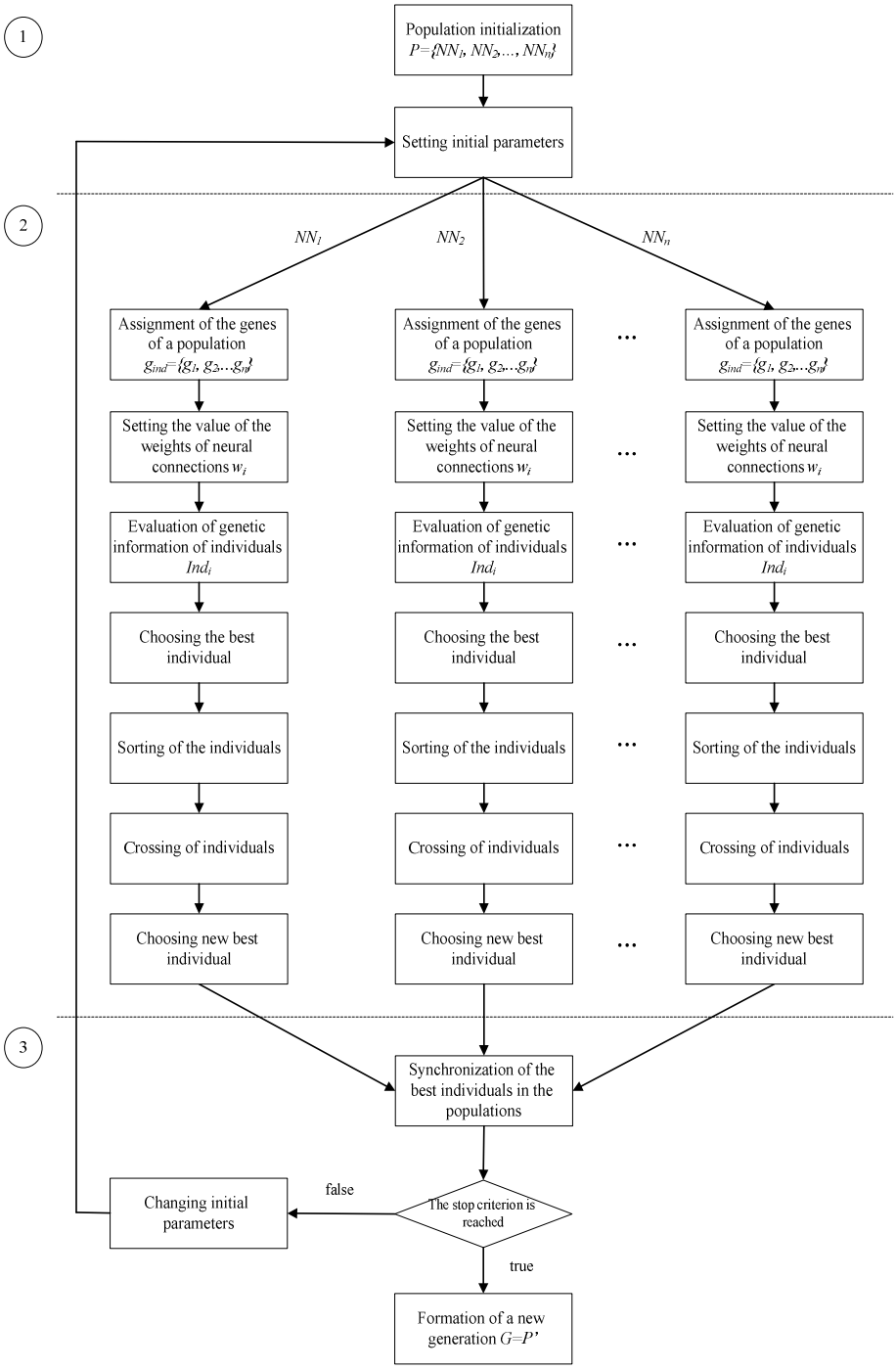

Fig. 3. Parallel genetic method for RNN synthesis

The proposed parallel method for RNN synthesis can be applied both on MIMD-systems [22] (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 [23]:

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: Parkinson's Disease Classification Data Set [24]. The data used in this study were gathered from 188 patients with PD (107 men and 81 women) with ages ranging from 33 to 87 (65.1Â±10.9). The data used in this study were gathered from 188 patients with PD (107 men and 81 women) with ages ranging from 33 to 87 (65.1Â±10.9) at the Department of Neurology in Cerrahpasa Faculty of Medicine, Istanbul University. The control group consists of 64 healthy individuals (23 men and 41 women) with ages varying between 41 and 82 (61.1Â±8.9). During the data collection process, the microphone is set to 44.1 KHz and following the physicians examination, the sustained phonation of the vowel was collected from each subject with three repetitions. Table 1 shows the main characteristics of the data sample.

Table 1. Main characteristics of the Parkinson's Disease Classification Data Set

Criterion	Characteristic	Criterion	Characteristic
Data Set Characteristics	Multivariate	Number of Instances	756
Attribute Characteristics	Integer, Real	Number of Attributes	754

6 The results analysis

In the Fig. 4 and 5 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 [25], 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.

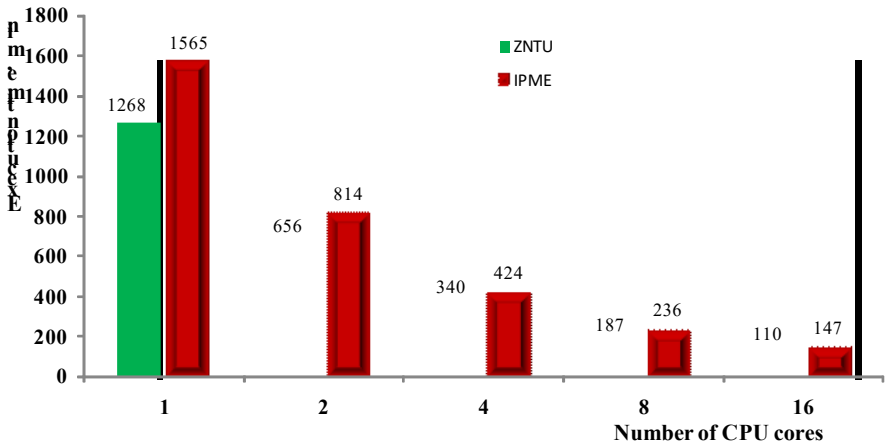

Fig. 4. Dependence the execution time of the proposed method to the number of involved cores of IPME cluster and ZNTU the computing system

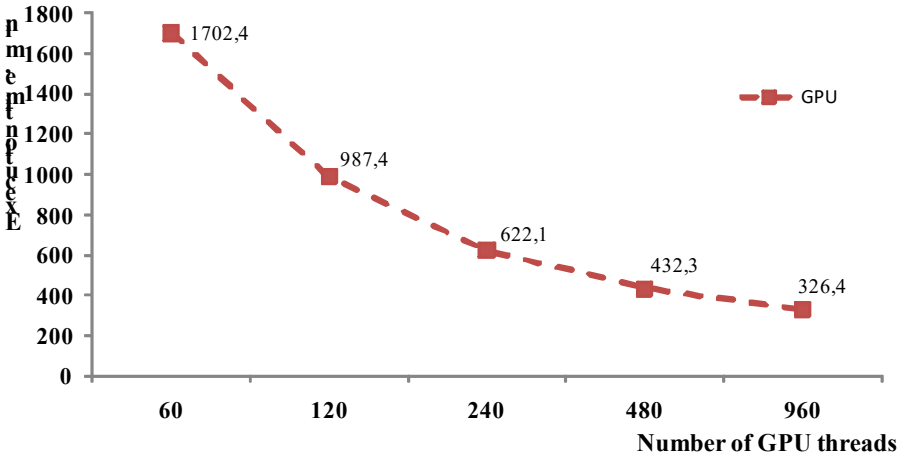

Fig. 5. Dependence the execution time of the proposed method to the number of GPU cores involved

The speedup graphics of calculations on a cluster IPME, ZNTU computing system and the GPU are shown in Fig. 6 and 7.

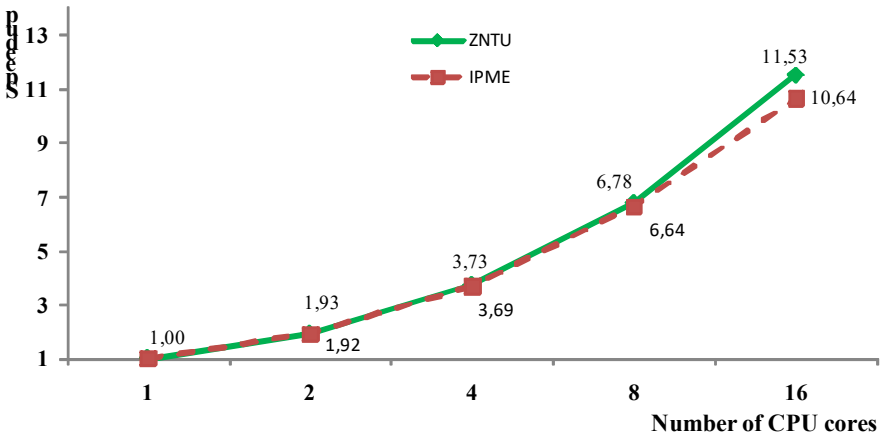

Fig. 6. The speedup graphics of calculations on a cluster IPME and ZNTU computing system

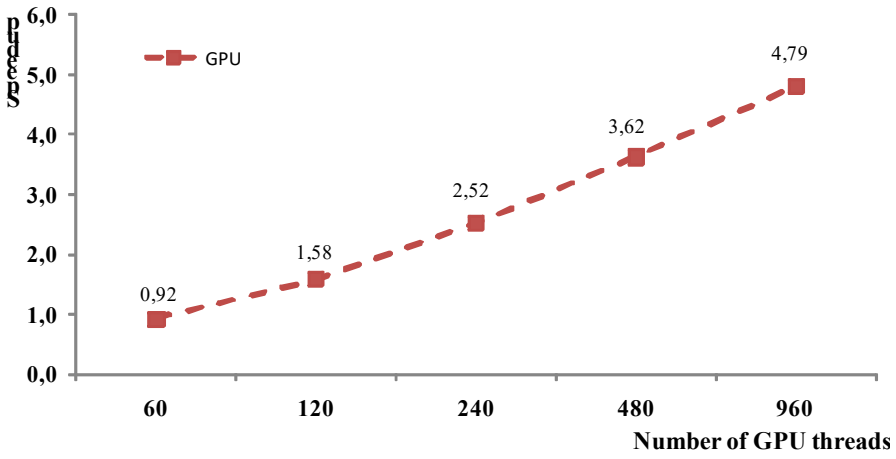

Fig. 7. The speedup graphics of calculations on a GPU

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. 8, 9), 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. 10. 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.

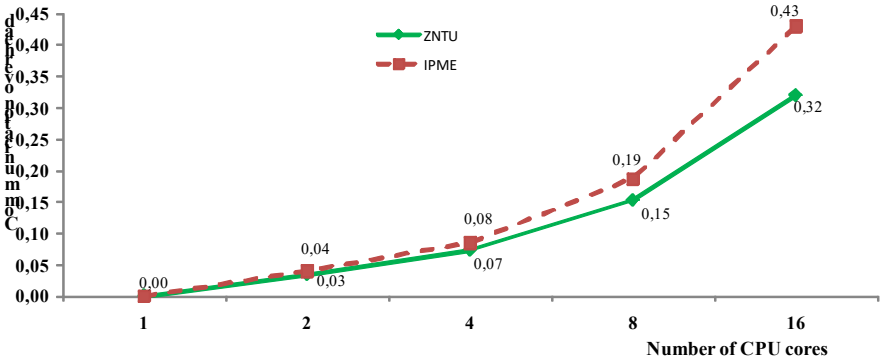

Fig. 8. Communication overhead performing the proposed method to the number of cores involved of IPME cluster and ZNTU the computing system

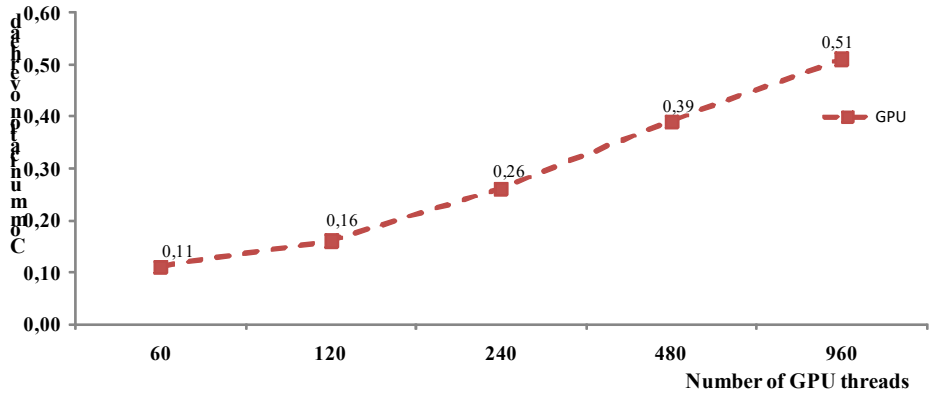

Fig. 9. Communication overhead performing the proposed method to the number of GPU cores involved

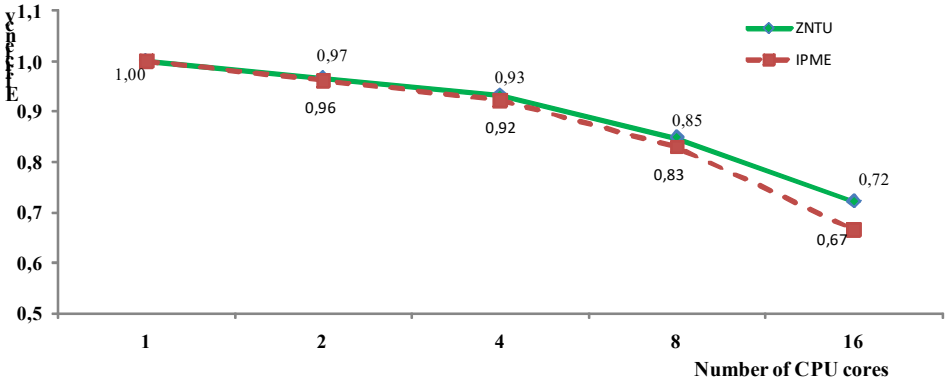

Fig. 10. The efficiency graph of IPME and ZNTU computing systems when executing the proposed method

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. 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.

A model of parallel genetic method of RNS synthesis is proposed, which in comparison with the sequential implementation significantly speed up the synthesis proc-

ess. 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. Volchek, Y.A., Shyshko, V.M., Spiridonova, O.S., Mokhort, T.V.: Position of the model of the artificial neural network in medical expert systems. Juvenis scientia (9), 4–9. Scientia, Saint Petersburg (2017).
2. 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, pp. 67–73 (2017). doi: 10.15587/1729-4061.2017.110142
3. 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), pp. 1-13 (2018). dblp key: conf/iddm/PerovaBSKR18
4. 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, pp. 369–376 (2006). doi: https://doi.org/10.1145/1143844.1143891.
5. 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, pp. 304-307, Lviv-Slavske, Ukraine (2018). doi:10.1109/TCSET.2018.8336208
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. 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, pp. 939–943, Kyiv (2017). doi: 10.1109/UKRCON.2017.8100388

10. 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, pp. 48–54 (2018). doi: 10.15587/1729-4061.2018.126578.
11. Handa, A., Patraucean, V.: Backpropagation in convolutional LSTMS. University of Cambridge Cambridge, pp. 1–5 (2015).
12. Boden M.: A guide to recurrent neural networks and backpropagation. Halmstad University, pp. 1–10 (2001).
13. Guo, J.: BackPropagation Through Time, pp. 1–6 (2013).
14. Yue, B., Fu, J., Liang, J.: Residual Recurrent Neural Networks for Learning Sequential Representations. Information, 9, 56 (2018).
15. 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), pp. 3–17 (2015).
16. 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, pp. 452-457, (2017). DOI: 10.1109/DT.2017.8024332
17. Balakrishan K., Honavar V.: Properties of Genetic Representation of Neural Architectures. Iowa State University (1995).
18. Whitley D. Genetic Algorithms and Neural Networks. Genetic Algorithms in Engineering and Computer Science, pp. 203-216, (1995).
19. Mitchell M. An introduction to Genetic Algorithm. MIT Press (1996).
20. Hochreiter, S., Schmidhuber, J.: Long Short–Term Memory. Neural Computation, vol. 9. issue 8, pp. 1735–1780 (1997).
21. 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.
22. Skillicorn, D.: Taxonomy for computer architectures. Computer (21), pp. 46-57 (1988). doi: 10.1109/2.86786.
23. 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, pp. 8634-8638, (2018). doi: 10.3923/jeasci.2018.8634.8638.
24. Parkinson's Disease Classification Data Set, https://archive.ics.uci.edu/ml/datasets/ Parkinson%27s+Disease+Classification
25. Intel Turbo Boost Technology 2.0, https://www.intel.com/content/www/us/en/architecture-and-technology/turbo-boost/turbo-boost-technology.html