UDC 004.4 https://doi.org/10.15407/pp2020.02-03.313

NEURAL NETWORKS' LEARNING PROCESS ACCELERATION

L. Katerynych, M. Veres, E. Safarov

This study is devoted to evaluating the process of training of a parallel system in the form of an artificial neural network, which is built using a genetic algorithm. The methods that allow to achieve this goal are computer simulation of a neural network on multi-core CPUs and a genetic algorithm for finding the weights of an artificial neural network. The performance of sequential and parallel training processes of artificial neural network is compared.

Key words: multi-core CPUs, artificial neural networks, artificial neuron, genetic algorithm, selection, crossing, mutation, parallel computing.

Дане дослідження присвячене розгляду процесу навчання паралельної системи у вигляді штучної нейронної мережі, побудованої за допомогою генетичного алгоритму. Методами, які дозволяють досягти поставленої в роботі мети, є комп'ютерне моделювання нейронної мережі на багатоядерних центральних процесорах та генетичний алгоритм для знаходження ваг штучної нейронної мережі. Наведено порівняння продуктивності послідовного та паралельного процесів навчання штучної нейронної мережі.

Ключові слова: багатоядерні центральні процесори, штучні нейронні мережі, штучний нейрон, генетичний алгоритм, відбір, схрещування, мутація, паралельні обчислення.

Данное исследование посвящено рассмотрению процесса обучения параллельной системы в виде искусственной нейронной сети, построенной с помощью генетического алгоритма. Методами, которые позволяют достичь поставленной в работе цели, являются компьютерное моделирование нейронной сети на многоядерных центральных процессорах и генетический алгоритм для нахождения весов искусственной нейронной сети. Приведено сравнение производительности последовательного и параллельного процессов обучения искусственной нейронной сети.

Ключевые слова: многоядерные центральные процессоры, искусственные нейронные сети, искусственный нейрон, генетический алгоритм, отбор, скрещивание, мутация, параллельные вычисления.

Introduction

The main task is to build an effective artificial neural network. It can be divided into two sub-tasks:

1. Selection of an appropriate structure (architecture, topology) for the artificial neural network.
2. Weighting for neurons of the artificial neural network.

The research tools are Microsoft Visual Studio Community Edition 2019 – an integrated development environment that supports several programming languages, including the C++ programming language.

The Neural Network and the Genetic Algorithm

The Concept of an Artificial Neural Network. Artificial neural networks (ANNs) are computational systems inspired by biological neural networks [1] that are part of the brain of living beings. Similar systems learn (gradually improving their productivity on tasks) by examining different samples, generally without special programming procedure for the task. The main component of any neural network is a neuron [1].

An Artificial Neuron. The artificial neuron (AN) is a simplified model of a biological neuron. From a mathematical point of view, an AN is represented as a function of a single argument. This function is commonly referred to as the activation function or transfer function. As an argument, it can take a linear combination of input signals, maximum or minimum of input signals, and so on. Structurally, an AN is composed of next parts (fig. 1) [3]:

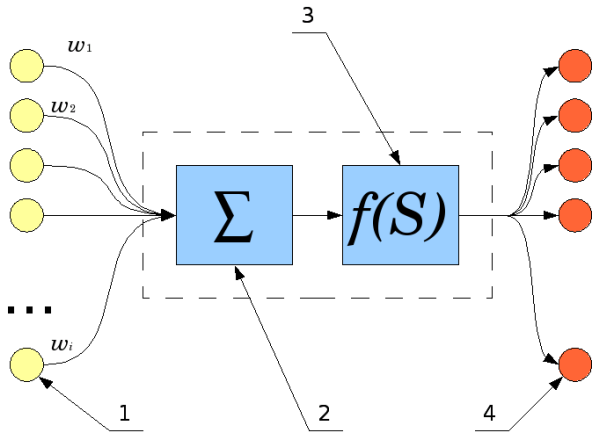

Fig. 1. The structure of an AN. w_1, w_2, …, w_i are synaptic scales

1. ANs, the output signals of which are input to the given neuron.
2. The adder of the input signals. It can find a linear combination of input signals, sum, maximum, minimum, average, square of the norm between input vectors and synaptic weights (radial function), etc.
3. The calculator of the activation function. Calculates some function f(x) from the value at the output of the adder. In general case, it can be arbitrary.
4. ANs, the inputs of which receive the signal from this neuron.

The Structure of an Artificial Neural Network. Typically, ANs in the ANN are grouped into layers. In general, there are no restrictions on the number and structure of layers, but they usually distinguish one input layer, one output layer, and several (or even none) intermediate (or hidden) layers (fig. 2).

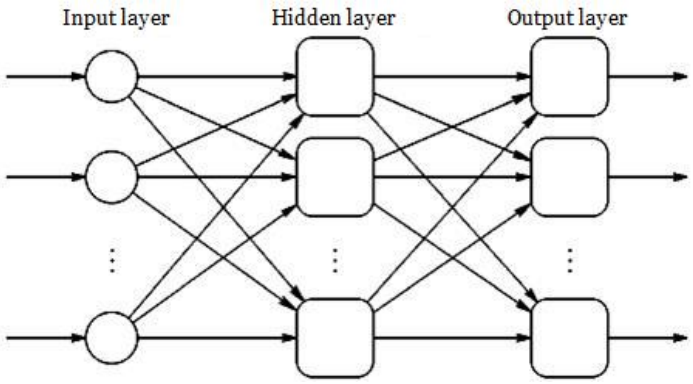

Fig. 2. A typical layer structure of an ANN

The Creation of Artificial Neural Networks. The creation of an ANN can be divided into two steps:

1) hoice of type (topology, architecture) of the ANN;
2) the training process of the ANN.

The first step is to determine the following:

- the number and types of input and output neurons;
- the number and type of neurons in the intermediate layers that will be used in the creation of the ANN;
- the type of connections of an AN.

In the first phase the type of the ANN is usually selected from one of the well-known architectures or used a genetic algorithm to move this task to a computer. In the second phase, it is necessary to "train" the ANN and to select right values of synaptic weights [5].

The training process itself can be represented in a cycle view with the following steps (fig. 3):

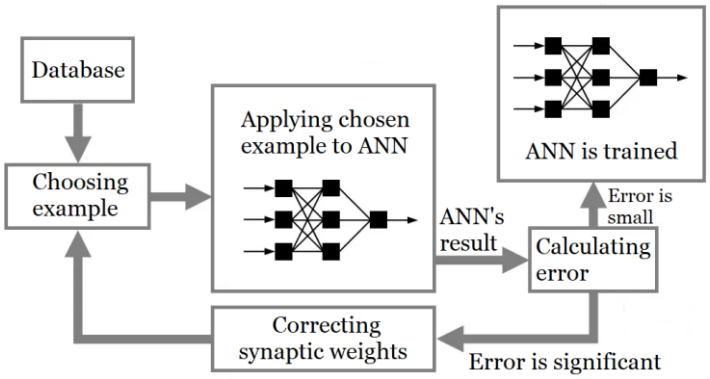

Fig. 3. General algorithm of ANN training

1. Choosing an example from the database. An example is the input and the expected output.
2. Applying the selected example to the ANN.
3. Error calculation.
4. Correction of synaptic weights of the ANN.

The Concept of the Genetic Algorithm. The genetic algorithm (GA) is an evolutionary algorithm used to solve optimization and modelling problems by sequentially selecting, combining, and varying the required parameters using mechanisms that resemble biological evolution [9].

Steps of the Genetic Algorithm. To solve a problem using the GA, the problem must be encoded so that its solution can be represented as an array of information – like composition of the chromosome. In addition, it is necessary

to define the fitness function, which will show if obtained GA solutions satisfy the optimal solution. This is a simulation of an evolutionary process that continues over several life cycles (generations) until the stop condition is fulfilled.

Thus, it is possible to distinguish the following steps of GA iteration [9] (fig. 4):

1. Creation of the original population.
2. Calculating fitness functions for the population (estimation).
3. Select individuals from the current population (selection).
4. Crossing of selected individuals and mutation.
5. Formation of a new generation.
6. If stop condition is not fulfilled yet, go to step 2.

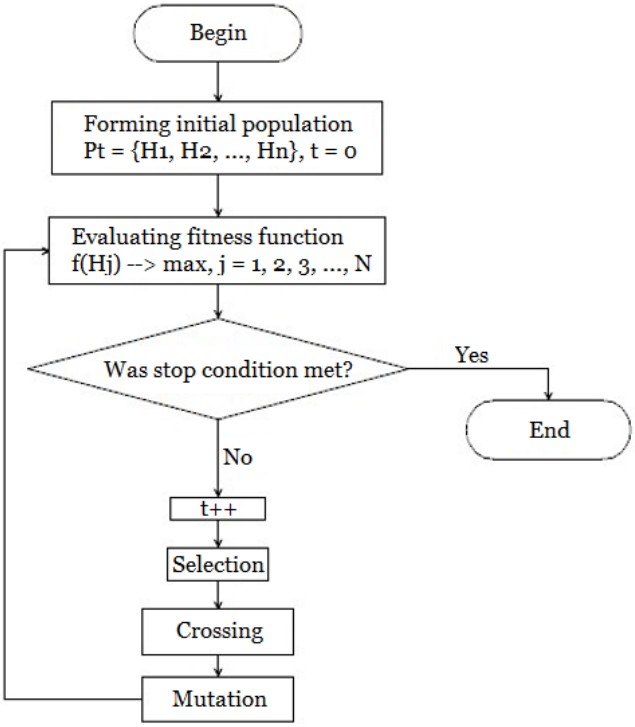

Fig. 4. Steps of the GA

Parallel Computing in Neural Networks

Parallel computing means software systems are developed as a set of computing processes that run simultaneously (in parallel) and can communicate mutually.

Parallel Computing at the Training Phase. In such a parallel computing organization, each thread calculates its ANN configuration different from the configurations of the other threads (Fig. 5).

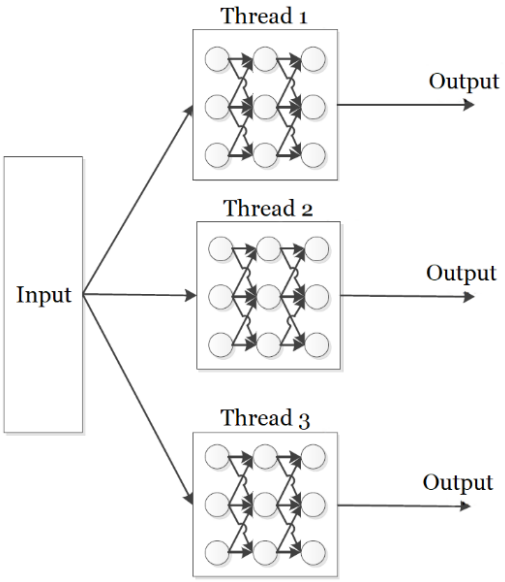

Fig. 5. Scheme of parallel computing at the training phase

Accordingly, several configurations of the ANN can be calculated at one time lapse on the same input data, which can give a linear increase in the acceleration of the training phase. This is possible due to the fact, that no interactions take place between processes.

Parallel Computing at the Neuron or Small Group of Neurons Level. With this kind of computation organization, each thread calculates the output of one neuron, or all neurons in a group. Typically, neurons of the same layer are split between threads (fig. 6).

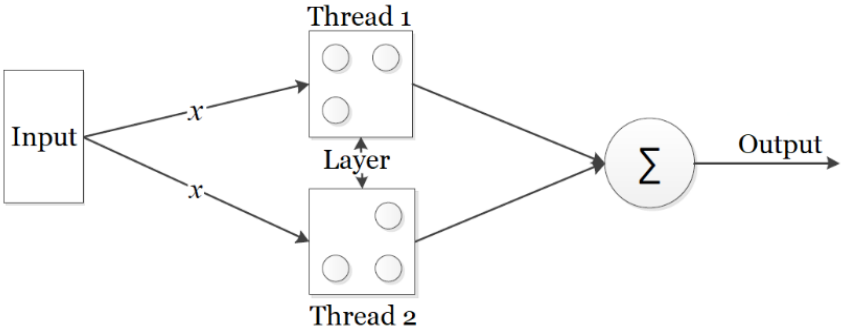

Fig. 6. Scheme of parallel computing at the level of a group of neurons

Accordingly, at one time lapse the output values of several neurons are found at once. In this case attention must be paid to synchronize between neurons of the current and previous layers (or between neurons of the current layer and the previous layer as a whole).

Comparison of Productivity of Sequential and Parallel Neural Networks

Problem Statement for the Neural Network. The problem, for which the ANN is built and trained:

A raster black and white image of a figure 40 by 40 pixels is given. It is necessary to resolve what this figure is. It is needed to build and train an ANN that accepts a vector of 1600 elements (by the number of pixels in an image) and outputs a vector of 10 elements. If the input is presented with an image of some digit *D*, then the maximum value must be achieved at the exit of number *D*. If an image other than a digit is submitted to the input, the ANN produces some incomprehensible result.

Description of the Neural Network's Structure. As an architecture, for this example, the direct propagation ANN with two hidden layers is considered. The number of layers and neurons in each layer is shown in fig. 7. The AN of each successive layer only connects with the neurons of the previous layer.

In the input vector, all values are between zero and one.

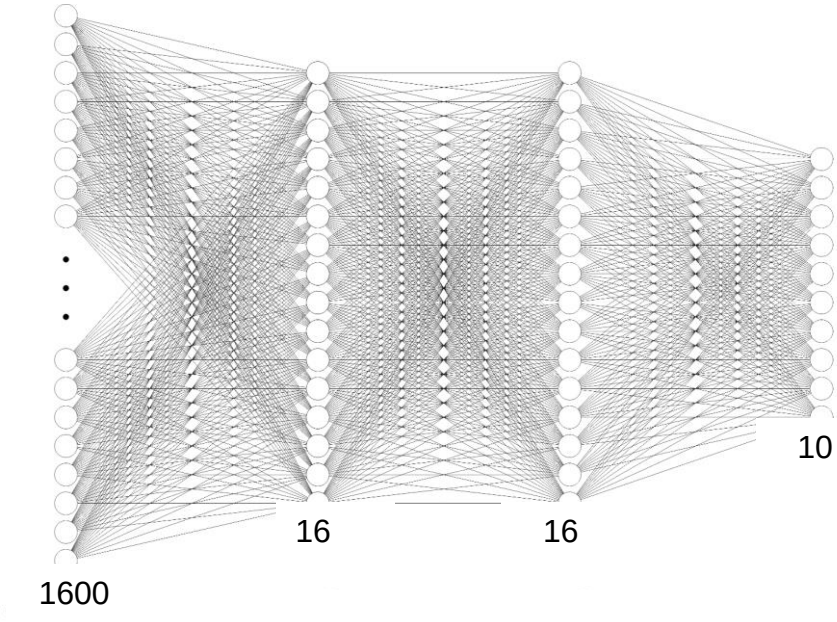

Fig. 7. The structure of the ANN for solving the above problem

Problem Statement for the Genetic Algorithm. The algorithm of training for the above ANN should be the GA. The solution should be coded as a sequence of synaptic weights of all neurons in the hidden and output layers. In

fact, the chromosome for the ANN is the sequence of real numbers. The function of adaptation is the sigmoidal function of the negative square in the norm of the difference between the expected and the output vectors. The initial population is 500 individuals. We take the fraction *s* of survivors equal to 0.5.

Two values are calculated: the fitness function f_i for each individual in a population of *N* ones and the sum of all fitness functions *F*. Then for all values of f_i the relation P_i is given by the following formula:

$$P_i = \frac{f_i}{F}, i = 0,1,\ldots,N-1. \quad (1)$$

Then the interval [0; 1] is divided into *N* intervals, with the length of the *i*-th interval being equal to the value of P_i. After that there is the generation of random numbers *z* from 0 to 1. If *z* falls into *i*-th interval, the individual enters the set of individuals that survived *H'*, and is removed from the set *H*. All values of P_j are recalculated as follows:

$$P_j = \frac{P_j}{P_i}, j = 0\ldots N, j \neq i, \quad (2)$$

after which the *i*-th interval of length P_i is removed from consideration. This continues until the set *H'* contains *sN* individuals.

The crossing operator is an intermediate recombination operator. For the given operator, the value of parameter *d* is set to 0.25. For reproduction, the individuals will be selected from the proportion of survivors in the current step. The crossing will occur until the number of individuals in the current population is equal to the number of individuals in the initial population, that is, 500 individuals.

The proportion of mutants *m is* chosen equal to 0.1.

For each mutant, the mutation oscillator randomly alters all synaptic weights of 5 randomly selected neurons. This process is repeated *mN* times.

To limit the number of iterations of learning, 1000 generations are taken into the consideration.

Performance Comparison of Serial and Parallel Neural Networks. Since the idea of GA involves random changes in synaptic weights, and these changes must occur once per cycle, the method and parallelization of the ANN at the training sample and layer levels in this case are not suitable. Parallelization method at one neuron's weights requires a significant amount of resources for parallel computing (multicore CPU), so this approach is inappropriate and will not be considered.

Description of the Testing Equipment

The tests were performed on a computer with the following equipment:

- processor Intel Core i7-6700HQ with 4 cores, 8 threads and 2.6 GHz frequency;
- RAM capacity of 8 GB;
- hard drive with a speed of 7600 rpm.

Duration of the Sequential Version of the Neural Network

For the ANN, a sigmoidal function from the negative square of the norm of the difference between the expected and the output vectors was chosen as the fitness function for the test. The performance of a sequential version of the ANN, after completing one iteration of GA training, considering the first 100 iterations of learning is in fig. 8.

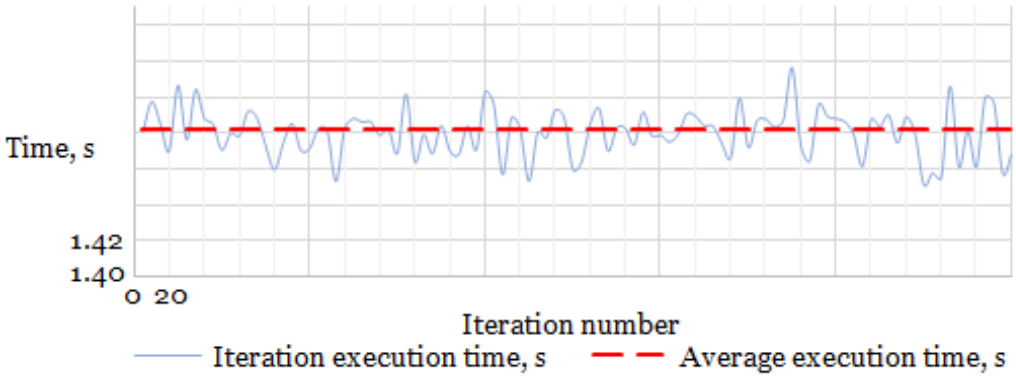

Fig. 8. The duration of the iteration, in the sequential calculation of the ANN

On average, it takes 1.48 seconds to perform a single GA iteration, a sequential version of the ANN.

Duration of Parallel Version with Phase-Level Parallelization

The performance of the parallel ANN version is different. The parallelization level is the training itself. One iteration of GA training was executed. Considering the first 100 iterations of training the results are in fig. 9:

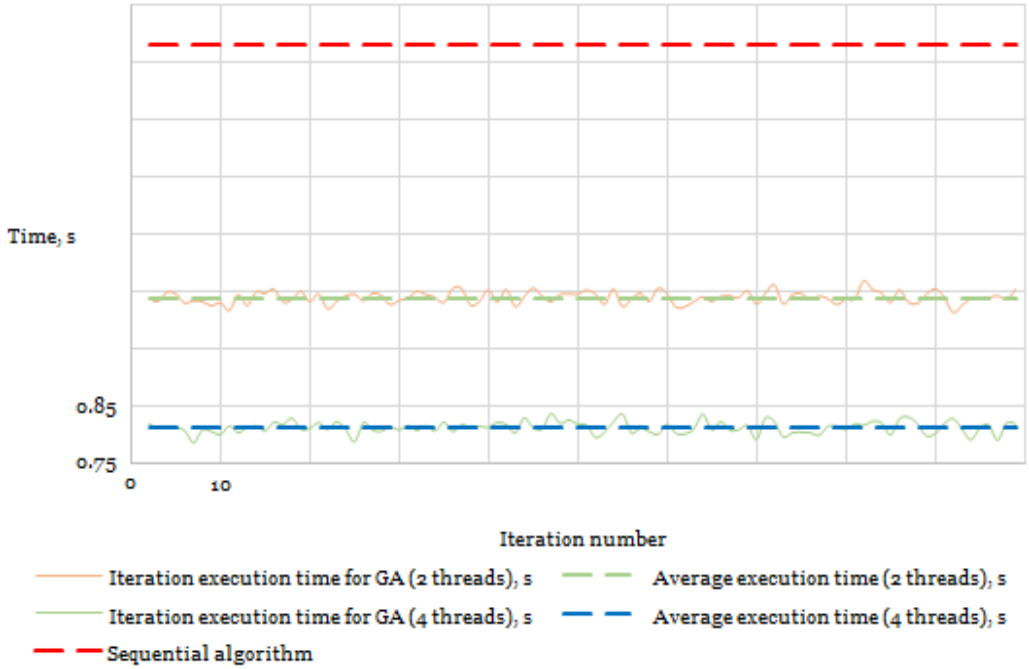

Fig. 9. Comparison of the duration of iteration, when splitting the ANN calculations at the learning level into 2 and 4 threads

When splitting the ANN calculations by this method, 1.04 seconds are spent on 2 threads to perform one GA iteration.

When splitting ANN calculations by this method, 0.81 seconds are spent on 4 threads to perform one GA iteration.

The acceleration and efficiency for this parallel version of ANN (fig. 10):

– 2 threads:

$$S_2 = \frac{1,48}{1,04} = 1,42\,, \tag{3}$$

$$E_2 = \frac{1,42}{2} = 0,71\,; \tag{4}$$

– 4 threads:

$$S_4 = \frac{1,48}{0,81} = 1,83\,, \tag{5}$$

$$E_4 = \frac{1,83}{4} = 0,46\,. \tag{6}$$

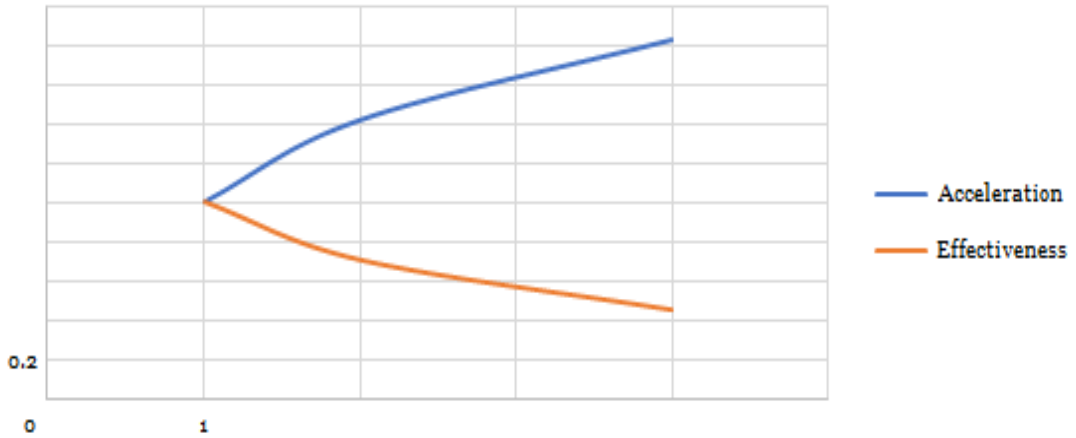

Fig. 10. The Parallelization Method at the Learning Level

Duration of the Parallel Version with the Parallelization at the Level of Groups of Neurons. The results at the level of groups of neurons are in fig. 11.

Fig. 11. Comparison of the duration of the iteration, when the computation in the ANN is at the level of groups of neurons and is split into 2 and 4 threads

When splitting ANN calculations by this method, 1.08 seconds are spent on 2 threads to perform one iteration of GA.

When splitting ANN calculations by this method, 0.86 seconds are spent on 4 threads to perform one iteration of GA.

The acceleration and efficiency for this parallel version of ANN (fig. 12):

– 2 threads:

$$S_2 = \frac{1,48}{1,08} = 1,37\,, \tag{7}$$

$$E_2 = \frac{1,37}{2} = 0,69\,; \tag{8}$$

– 4 threads:

$$S_4 = \frac{1,48}{0,86} = 1,72\,, \tag{9}$$

$$E_4 = \frac{1,72}{4} = 0,43\,. \tag{10}$$

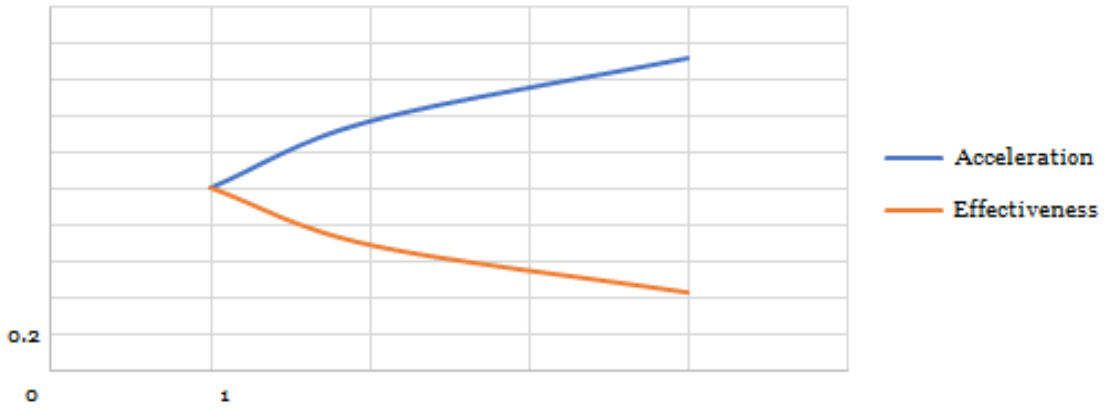

Fig. 12. The acceleration and efficiency of GA iteration in parallelization at the level of groups of neurons

Evaluation of the Serial Part of the Iteration Cycle. Based on the law of Amdal:

$$S_p = \frac{1}{\alpha + \frac{1-\alpha}{p}}, \tag{11}$$

where α – sequential part of the algorithm, $\alpha \in (0, 1)$, p – number of threads, an estimate of the sequential part of the algorithm for each of the above methods of parallelization can be found by the formula:

$$\alpha = \frac{1}{(p-1)}\left(\frac{p}{S_p} - 1\right). \tag{12}$$

Parallelization at the level of learning:
– 2 threads:

$$\alpha = \frac{2}{S_p} - 1 = \frac{2}{1,42} - 1 = 0,41; \tag{13}$$

– 4 threads:

$$\alpha = \frac{1}{3}\left(\frac{4}{S_p} - 1\right) = \frac{1}{3}\left(\frac{4}{1,83} - 1\right) = 0,40. \tag{14}$$

Parallelization at the level of groups of neurons:
– 2 threads:

$$\alpha = \frac{2}{S_p} - 1 = \frac{2}{1,37} - 1 = 0,46; \tag{15}$$

– 4 threads:

$$\alpha = \frac{1}{3}\left(\frac{4}{S_p} - 1\right) = \frac{1}{3}\left(\frac{4}{1,72} - 1\right) = 0,45. \tag{16}$$

Although in both cases the sequential part of the iteration of the training does not exceed the volume of the parallel part, they are comparable. Therefore, it is essential to accelerate the learning with the genetic algorithm, only by splitting the calculations of the output vectors of neural networks into several threads.

Conclusions

The goal was achieved, the features of the construction and training of the neural network using the genetic algorithm were considered and the acceleration of learning was analysed by the implementation of an artificial neural network as a parallel system.

In the case of GA, the synaptic scales of the ANN can be changed only once for the iteration of the training of each ANN, so parallelization at the training sample level and at the layer level does not make sense.

The parallelization of the learning phase makes it possible to accelerate the execution of the parallel part of the algorithm almost linearly with the serial version, due to the lack of information interactions between the threads.

Parallelization at the level of a neuron, or a small group of them, requires synchronization of computations between neurons of different layers and a considerable amount of resources for parallel computations, provided that many neurons themselves are in the ANN. It shows significant but not linear acceleration compared to the sequential version.

References

1. Chapter 2. Introduction [Electronic resource] / Peter Radko: Portal “Neural networks”.
2. Kononyuk A. NEURAL NETWORKS AND GENETIC ALGORITHMS / A. Kononyuk - Kyiv: Korniychuk, 2008.
3. Chapter 3. Fundamentals of INS [Electronic resource] / Peter Radko: Portal “Neural networks”.
4. Classification of neural networks [Electronic resource]: Artificial Intelligence Portal Project.
5. Hawkins, Douglas M. The problem of overfitting. *Journal of Chemical Information and Computer Sciences* - 2004 - 4 4.1
6. Ting Qin, et al. A CMAC learning algorithm based on RLS. *Neural Processing Letters* 2004. 19. 1
7. Crick F. The recent excitement about neural networks *Nature.* 1989. 337
8. Edwards C. Growing Pains for Deep Learning. *Communications of the ACM.* 2015. Vol. 58, Issue 7

9. Panchenko T. Genetic algorithms / Panchenko T. - Ed. Astrakhan University House, 2007.
10. Andrews G. Fundamentals of Multithreaded, Parallel, and Distributed Programming: Trans. with English. - M.: Williams House, 2003.
11. Bogachev K. Fundamentals of parallel programming / Bogachev K. - M.: BINOM. Lab. Knowledge, 2003.

Література

1. Chapter 2. Introduction [Electronic resource] / Peter Radko: Portal "Neural networks".
2. Kononyuk A. NEURAL NETWORKS AND GENETIC ALGORITHMS / A. Kononyuk - Kyiv: Korniychuk, 2008.
3. Chapter 3. Fundamentals of INS [Electronic resource] / Peter Radko: Portal "Neural networks".
4. Classification of neural networks [Electronic resource]: Artificial Intelligence Portal Project.
5. Hawkins, Douglas M. The problem of overfitting. *Journal of Chemical Information and Computer Sciences* - 2004 - 4 4.1
6. Ting Qin, et al. A CMAC learning algorithm based on RLS. *Neural Processing Letters* 2004. 19. 1
7. Crick F. The recent excitement about neural networks *Nature.* 1989. 337
8. Edwards C. Growing Pains for Deep Learning. *Communications of the ACM.* 2015. Vol. 58, Issue 7
9. Panchenko T. Genetic algorithms / Panchenko T. - Ed. Astrakhan University House, 2007.
10. Andrews G. Fundamentals of Multithreaded, Parallel, and Distributed Programming: Trans. with English. - M.: Williams House, 2003.
11. Bogachev K. Fundamentals of parallel programming / Bogachev K. - M.: BINOM. Lab. Knowledge, 2003.

Received 02.03.2020

About authors:

Katerynych Larysa,
PhD in Computer Science,
Associate Professor of Taras Shevchenko National University of Kyiv.
Publications in Ukrainian journals – 19.
Publications in foreign journals – 1.
https://orcid.org/0000-0001-7837-764X,

Veres Maksym,
PhD in Computer Science,
Associate Professor of Taras Shevchenko National University of Kyiv.
Publications in Ukrainian journals – 15.
Publications in foreign journals – 1.
https://orcid.org/0000-0002-8512-5560,

Safarov Eduard,
student of Faculty of Computer Science and Cybernetics,
Taras Shevchenko National University of Kyiv.
https://orcid.org/0000-0001-9651-4679.

Authors' place of work:

Taras Shevchenko National University of Kyiv,
03680, Kyiv-187,
Akademician Glushkov Avenue, 4d.
Phone: 050 537 6699.
E-mail: katerynych@gmail.com