COMPUTER SCIENCES AND INFORMATION TECHNOLOGIES

UDC 629.735.33-519(045)
DOI:10.18372/1990-5548.66.15222

[1]O. I. Chumachenko,
[2]M. O. Liubachenko

OPTIMAL GENETIC ALGORITHM SELECTION FOR DEEP NEURAL NETWORK SETTINGS

[1,2]Technical Cybernetic Department, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute," Kyiv, Ukraine
E-mails: [1]chumachenko@tk.kpi.ua ORCID 0000-0003-3006-7460, [2]ejie1997@gmail.com

***Abstract*—*The problem of construction of deep neural networks with the use of genetic algorithms is considered. The problem of structural-parametric synthesis with creation of neural networks is defined. The main purpose of the study is to find a deep neural network that is optimal for solving urgent problems. The classification problem is chosen as the urgent problem to solve. Also the classification of genetic algorithms is given, which are used as a basis for establishing the parameters of deep neural networks A system for the optimal tuning of the parameters of deep neural networks is proposed, which includes a two-stage algorithm. At the first stage of the algorithm, a multicriteria genetic algorithm is selected from a set of possible ones (genetic algorithm of vector estimation, genetic algorithm of Fonseca and Fleming, genetic algorithm of Pareto approximation with niche, genetic algorithm of sorting without dominance, genetic algorithm of Pareto force, genetic algorithm of Pareto-2 force) that best fits the given training sample. At the second step, the problem of structural-parametric synthesis of a neural network is solved according to the criteria of accuracy and complexity. As a result of training, the values of the neural network parameters are found, such as: the number of layers, the number of neurons in each layer, the values of the weight coefficients.The modeling of the proposed system is carried out. The results of modeling, comparison of results with similar software packages are presented. The obtained results show the possibilities of wide use.*

Index Terms—Multiobjective genetic algorithm; deep neural networks; structural-parametric synthesis; optimal selection.

I. INTRODUCTION

Modern systems that require solutions to problems of high complexity are keep using artificial neural networks (ANN) [1] – [12]. Such problems include solving the problem of classification, forecasting, clustering, decision making, approximation, data analysis, optimization. The use of ANN can be widely found in many areas – construction (for example, finding the optimal design parameters), medicine (disease recognition), economics (forecasting the exchange rate) and others. But when using ANN there is a problem of decision-making on the type of neural network and its parameters, depending on many criteria.

This article shows an algorithm for determining the best genetic algorithm for learning a neural network with the definition of its structure (number of layers, number of neurons in layers) and values of weights, which differs from those known in that in order to improve the quality of learning the best of the algorithms.

As a result, we get a system that increase the efficiency of solving problems of neural networks due to a multiobjective optimization system – the optimality of the parameters reduce the complexity of learning neural networks while obtaining optimal accuracy.

It can be used in practice to improve neural network problem solving – it can be used to reduce resource use in problem solving.

II. PROBLEM STATEMENT

Structural-parametric synthesis [2] is a process as a result of which the structure of the object is determined and the parametric values of its constituent elements are found, so that the conditions of the synthesis task are fulfilled. If the synthesized object is optimal by any criteria, then the synthesis itself is optimal.

The used mathematical and computer models used for automation of structural-parametric synthesis of objects differ significantly from those models used for automation of parametric synthesis. It follows that if the structure of the object in the synthesis process does not change during parametric synthesis, then during the process of structural-parametric synthesis there is a change in the parameters of the object and a change in the structure.

Statement of the problem of structural-parametric synthesis in our case will determine the structure of the model and their parameters are two criteria (complexity and accuracy).

To solve the problem of structural-parametric synthesis in our case it is necessary to determine the model structure and its parameters, namely – the model complexity of the deep neural networks (DNN) and accuracy.

Let the maximum number of neurons be given A in a neural network constructed to approximate the dependence on the sample of source data $<X, Y>$ where $X = \{X_i\}$ – a set of values of characteristics (features) that describe the object or process; $Y = \{y_p\}$ – array of parameter values at the output in this sample; $X_i = \{X_{ip}\}$ this is the ith feature in the sample, $i = 1, 2, ..., L$; X_p is the value of the ith attribute for the pth sample, $p = 1, 2, ..., M$; y_p is the value of the predicted parameter for the pth instance; L is the total number of objects in the original set; m – number of samples.

Then the task of the structural-parametric synthesis is to find a model of the form $HC = HC(S, W, B)$, for which $\xi(HC, X, Y) \rightarrow \min$, while $S = S(L, A)$ is a matrix that determines the presence of synaptic connections between network elements (input functions, neurons); $W = W(S)$ is a matrix of weights corresponding to the ratios present in the network; $B = B(S)$ is the shift vector of network neurons; $\xi(HC, X, Y)$ is the criteria that determine the effectiveness of the model of the DNN to approximate the relationship between the set of parameters at the input – X and the corresponding vector of parameter values at the output – Y.

III. Overview of Genetic Algorithms

To date, there are many genetic algorithms [1], but the most popular are VEGA (vector genetic algorithm), FFGA (multi-purpose genetic algorithm) Fonseca and Fleming) or MOGA (also called a multi-purpose genetic algorithm), NPGA (Niche-Pareto genetic algorithm), SPEA and SPEA2 (Pareto force evolutionary algorithm).

1) VEGA. David Schaffer (1984) [3] extended Grefenstette's program GENESIS to include multicriteria functions. Schaffer's approach was to use an extension of the Simple Genetic Algorithm (SGA), which he called the Vector Genetic Algorithm (VEGA), and which differed from SGA only in selection. This operator was modified so that a number of subpopulations were generated for each generation, performing proportional selection according to each criterion. Thus, for a problem with the k criteria, subpopulations of the size of N / k are generated (assuming N is the total population size). These subpopulations will be mixed to obtain a new population of size N, which will be subject to GA, in which crossover and mutation operators are used in the usual way. In Figure 1 a structural representation of this process is shown.

The main advantage of this algorithm is its simplicity, ie this approach is quite easy to implement. Richardson and co-authors. (1989) [4] note that shuffling and merging of all subpopulations corresponds to the averaging of the custom components associated with each of the criteria. Because Schaffer used the proportional purpose of fitness, these components of fitness, in turn, were proportional to the criteria themselves. Thus, the obtained expected adaptation corresponds to a linear combination of goals, where the weights depend on the distribution of the population in each generation, as shown by Richardson and co-authors.

The main disadvantage of this is that when we have a concave compromise surface, certain points in the concave regions will not be found by this optimization procedure, in which we use only a linear combination of criteria, and it has been proven that regardless of the set of weights are used. Thus, the main weakness of this approach is its inability to produce Pareto-optimal solutions in the presence of non-convex search spaces

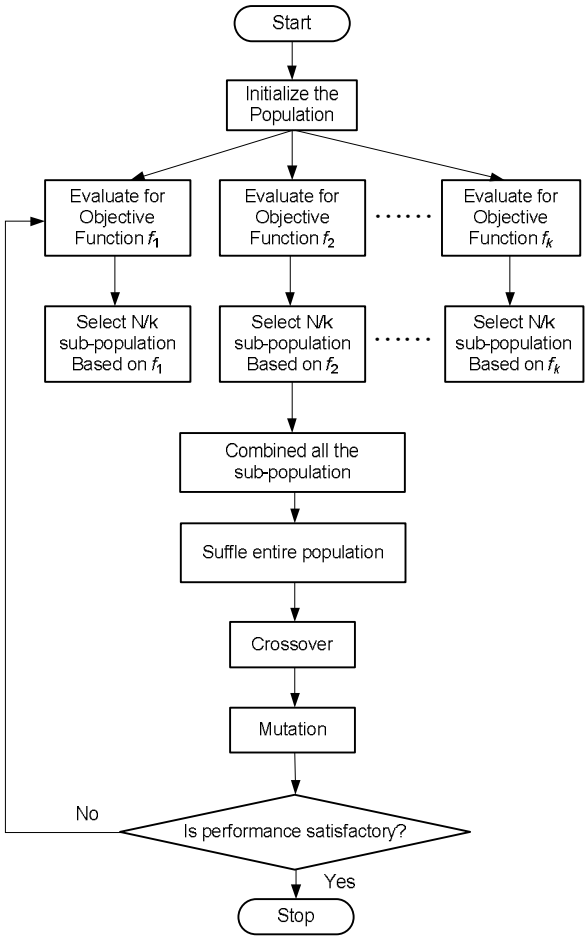

Fig. 1. VEGA algorithm

2) FFGA. Fonseca and Fleming (1993) [4] implemented Goldberg's proposal differently. First, let's discuss what Goldberg has to offer in terms of Pareto's rating.

Goldberg proposed a Pareto ranking scheme in (1989), where the solution *x* during generation t has a corresponding target vector x_u, and *n* is the size of the population, the rank of the solution is determined by the following algorithm.

FFGA algorithm is described below:

1. $curr\_rank = 1;\ m = n;$
2. While $n! = 0$ do
3. For $i = 1\ldots, m$ do
4. If x_u is non-dominated
$rank\ (x, t) = curr\_rank$ end
do
5. For $i = 1, \ldots, m$
6. do
7. If $rank(x, t) = curr\_rank\ Delete$ x from population $n = n - 1;$
8. The end of doing
9. $curr\_rank = curr\_rank + 1\ m = n$
10. end while

This means that the entire population is checked in the Pareto rating, and all individuals who do not dominate are assigned the rank "1". These individuals are then removed from the population with rank "1". All non-dominated individuals from the rest of the population are identified again and assigned the rank "2". Thus, the procedure continues until all decisions receive the required rank.

But in multiobjective genetic algorithms, the entire population is tested, and all individuals who do not dominate are assigned the rank of "1". Other individuals are classified by checking their dominance relative to the rest of the population as follows.

For example, an individual x_i in a generation *t* in which $p_i^{(t)}$ individuals predominate in the current generation. Its current position in the rank of individuals can be given by

$$\text{Rank}(x_i, t) = 1 + p_i^{(t)}.$$

Once the ranking procedure is complete, it is time to assign fitness to each individual. Fonseca and Fleming proposed two methods for determining fitness:

– rank based determination of fitness;

– methods of forming niches.

The appointment of fitness on the basis of rank is as follows:

– sort the population by rank;

– assign fitness by interpolating from best (rank "1") to worst (rank $n \le N$) in the usual way, according to a certain function, usually linear, but not required.

– On average, assess the level of fitness of individuals with the same rank so that they all participate at the same rate. This procedure maintains a constant adaptation of the global population, maintaining the appropriate selection pressure, as determined by the function used.

As Goldberg and Deb pointed out, this type of blocked adaptation is likely to lead to high selection pressures, which can lead to premature convergence. To avoid this, Fonseca and Fleming used the second method (i.e., the niche formation method) to distribute the population over the optimal Pareto region, but instead of exchanging parameter values, they used exchanging the values of the objective function.

The main advantage is that it is effective and relatively easy to implement. The effectiveness of this method strongly depends on the distribution coefficient σ_{share}. However, Fonseca and Fleming have developed a good methodology for calculating this value for their approach.

3) NPGA. Horn, Nafploitis, and Goldberg [1], [5] proposed an NPGA based on a Pareto dominance tournament and an exchange of equivalence classes.

Pareto-dominant tournament. Basically, it is a scheme of tournament selection based on Pareto dominance. In this scheme, a selection set of comparisons consisting of a certain number (t_{dom}) of individuals is randomly selected from a population at the beginning of each selection process. Two individuals are selected at random from the population for selection. Then each of the individuals is compared with each individual in the comparison set. If in one the set of comparison prevails, and in another is not present, the later is selected for reproduction. If neither or both are dominated by a set of comparisons, then we move on to the second technique.

Exchange of equivalence classes. Since both individuals are the same, ie either dominant or non-dominant, it is likely that they are in the same class of equivalence. So in this case we choose the "best fit" according to the following procedure.

We choose the radius of the niche σ_{share}, and according to this radius, the candidates who have the smallest number of individuals in the population are the most suitable. In the following Fig. 2 shows how this procedure works: here we maximize along the *x*-axis and minimize along the *y*-axis.

In this case, the set of candidates for selection does not exceed the set of comparison. Thus, in

terms of the number of niches, this shows that candidate 1 is the best fit. Here t_{dom} is selected only once for a certain generation t. After creating a new population, a genetic operator similar to other methods is used.

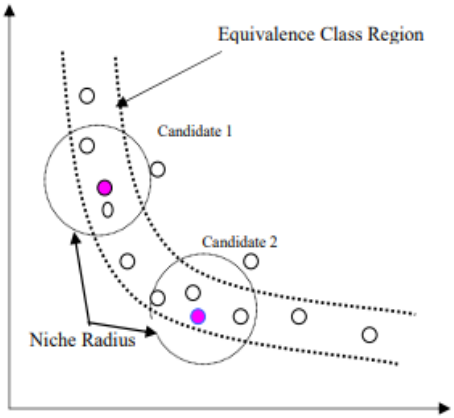

Fig. 2. Exchange of equivalence classes

NPGA algorithm is written below:

1. Initialization of population with the size n.
2. The choice of t_{dom} chromosome size is random from the current population.
3. Random selection of two chromosomes from the current population.
4. Selection of n individuals based on the following procedure: Compare two chromosomes with tdom for nondominance by the previous definition. If one is dominant and the other is not dominant, choose one that is not dominant. If both do not dominate or dominate, then choose the best chromosome (individual) by the method of niche formation.
5. Apply crossover and mutation to get a new population.
6. Check if the performance criteria are met, if not, go to step 2, otherwise go to step 7.
7. Stop.

Because this approach does not apply Pareto selection to the entire population, but only to its segment at each run, its main advantages are that it is very fast and that it creates good nondominant fronts that can be maintained for many generations.

The effectiveness of this method strongly depends on the partition coefficient (σ_{share}) and good tournament numbers (t_{dom}) and difficult to implement.

4) NSGA. NSGA [6] differs from a simple GA only in the form in which the selection operator is used. Operator crossover and dominant solutions are also important in order to obtain a good distribution of solutions in the optimal Pareto front. Adaptation is performed in two stages.

1. Assigning the same fictitious adaptability to all decisions of a certain level of dominance.
2. Application of exchange strategy.

We will now discuss the details of these two steps.

First, all decisions on the first nonpredominant front are assigned an adaptation equal to the population size. This becomes the maximum suitability that any solution can have for any population. Based on the sharing strategy, if a solution has many adjacent solutions on one front, its fictitious adaptability is reduced by a factor and the total fit is calculated. The factor depends on the number and proximity of neighboring solutions. After all decisions on the first front are assigned values of fitness, the lowest total value of fitness is determined.

After that, the persons who are on the second level of domination are assigned a fictitious adaptation, equal to the number less than the least total adaptation of the previous front. This ensures that no decision on the second front has a more common fit than any decision on the first front. This maintains pressure on the solution to lead to an optimal Pareto front. The method of sharing is again used among the persons of the second front, and the general adaptability of each person is revealed. This procedure is continued until all individuals have received general fitness.

After the fitness assignment method [7], [8], roulette selection (RWS) is used to select N individuals. Crossover and mutation are then used. Joint fitness is calculated as follows.

Given the set of n_1 decisions in the lth non-predominant front, each of which has a fictitious value f_1, the common procedure is performed as follows for each decision $i = 1, 2, 3, ..., n_1$.

1. Calculate the normalized measure of the Euclidean distance with a different solution in the lth nonpredominant front, as shown below:

$$d_{ij} = \sqrt{\sum_{p=1}^{p}\left(\frac{x_p^{(i)} - x_p^{(j)}}{x_p^{(u)} - x_p^{(s)}}\right)^2},$$

where P is the number of criteria in the problem. These parameters $x_p^{(u)}$ and $x_p^{(s)}$ are the lower and upper limits x_p.

Calculate the normalize.

2. The distance is compared with a predefined parameter and the following value of the sharing function is calculated:

$$sh\left(d_{ij}\right) = \begin{cases} 1 - \left(\dfrac{d_{ij}}{\sigma_{share}}\right)^2, & \text{if } d_{ij} \le \sigma_{share}, \\ 0, & \text{else.} \end{cases}$$

3. Incrementation *j*. If $j \le n_1$ so, go to step 1 and calculate $sh(d_{ij})$. If $j > n_1$, calculate the number of niches for the *i*th solution as follows:

$$m_i = \sum_{j=1}^{n_1} sh(d_{ij}).$$

4. Determine the fictitious suitability f_1 of the $\dot{f}_i$ solution in the *l*th nondominant front, to calculate the overall suitability, as follows:

$$\dot{f}_i = \frac{f_1}{m_i}.$$

This procedure continues for all *i* = 1, 2, 3, ..., n_1 and found accordingly. After that, the smallest value $f_i^{\min}$ of min from all in $\dot{f}_i$ *l*th nonpredominant front was found for further processing. The fictitious adaptation of the next nonpredominant front is determined by $f_{l+1} = f_i^{\min} - \varepsilon_1$ where ε_1 a small positive number.

The above sharing procedure requires a predefined parameter σ_{share}, which can be calculated as follows:

$$\sigma_{\text{share}} \approx 0.5 / \sqrt{q}^{1/p},$$

where *q* is the desired number of different Pareto-optimal solutions. Although the calculation σ_{share} depends on this parameter *q*, *q* = 10 has been shown to work well for many test tasks.

The main advantage of this method is that it can handle any number of criteria, and this makes the distribution in the space of parameter values instead of the value space of the object, which provides a better distribution of people and allows to obtain several equivalent solutions.

Some researchers note that this is ineffective when considered as the computational efficiency of the produced Pareto fronts. Another disadvantage is that it is more sensitive to σ_{share}.

5) SPEA, SPEA-2

General scheme of SPEA (Evolutionary force algorithm Pareto) [6], [8], [9]:

– initialization: generating the 1st set and creating an empty external Pareto-optimal set (archive);

– Pareto-optimal set updating. If the value of the Pareto-optimal set exceeds the specified limit, subsequent Pareto networks are destroyed by the clustering method. To reduce the Pareto set to a controlled size, the average algorithm of hierarchical clustering based on compounds is used. It performs an iterative combination of adjacent clusters to achieve the required number of groups;

– calculating the value of the fitness function for the external Pareto-optimal set and set of individuals;

– selection by means of tour selection: the population and the external set are combined, and any two persons are chosen at random. As for their fitness function, the best of them moves to the pool (mating pool). A pool is a collection of intersecting populations that undergo mutation and crossover operations to create a new population;

– the new population is produced by mutation and crossover operations;

– setting $t = t + 1$. If the stop criterion $(t > T)$ is not met, go to step 2; otherwise the members of the archive are represented as the optimal Pareto set.

SPEA differs significantly from its predecessors in that:

– the concept of Pareto dominance is used to assign scalar suitability to individuals;

– persons who do not dominate other members of the population of Gindi, are stored separately in a special external set (archive);

– to reduce the number of persons stored in the archive, clustering is carried out, which, in turn, does not affect the characteristics of persons acquired in the search process.

The uniqueness and advantages of the SPEA method is that:

– it combines the above approaches in one algorithm;

– the suitability of each individual of the population is determined only in relation to the persons of the external archive, regardless of whether the individuals of the population dominate each other;

– despite the fact that the "best" persons obtained in previous generations are stored in the external archives, they all participate in the selection;

– to prevent premature convergence, a special mechanism of niche formation is used, where the distribution of general suitability is carried out not in terms of the distance between individuals, but on the basis of Pareto dominance.

One of the disadvantages of SPEA is that most of the resources and time are spent on the clustering procedure, which provides support for population diversity.

When developing SPEA-2 [1] [10], the main goal was to eliminate the potential shortcomings of the predecessor (SPEA) and incorporate the latest results to create a powerful and modern multi-criteria evolutionary algorithm. The main differences between SPEA-2 and SPEA:

– an improved scheme for assigning a fitness function that takes into account each individual, how many people dominate him and how many people dominate others;

– the closest method of estimating the density of neighbors, which allows you to more accurately manage the search process;

– a new method of truncating archives, which guarantees the preservation of marginal solutions.

In general, one of the most important steps in SPEA-2 is to determine fitness function or fitness.

Determination of fitness [11]: performed on the basis of the concept of Pareto-dominance, the algorithm for calculating which (suitability *F*) for each individual from the population of P_t and the archive A_t has the following form.

1) Suppose we have a set of people who make up the P_t population and archive *B*, where each person is assigned a value $S(i) \in [0.1)$, is called "force" (which shows how many decisions it dominates), which is proportional to the number of members of the population $j \in P_t$, for which $f(i) \geq f(j)$, in the case of multicriteria optimization. The proportion is as follows:

$$S(i) = \frac{n}{N+1},$$

where *N* is the population size; *n* is the number of individuals that dominate under conditions $f(i) \geq f(j)$.

But the "strength" of each individual and the set of persons in the A_t archive and the set of populations P_t will be defined as the sum of "force" on persons and "force" taking into account the dominance encoded by criterion *i* above the criterion encoded by *j*:

$$S(i) = \left|\{j \,|\, j \in P_t + A_i \wedge i > j\}\right|,$$

where + stands for multiset union, the symbol ∧ stands for a conjuction operator and the symbol > corresponds to the Pareto dominance relation extended to individuals ($i > j$ if the decision vector encoded by *i* dominates the decision vector encoded by *j*).

2) Based on the value of $S(i)$ is calculated "raw" value of adaptation $R(i)$ of individual *i*, which is calculated by summing the "forces" of all individuals *j*, which dominate or weakly dominate by all criteria:

$$R(i) = \sum_{j \in P_t + A_t, j > i} S(j),$$

where P_t is the set of individuals of the population A_t a set of persons of the archive.

3) Density estimation method is an adaptation of the method of the *k*th nearest neighbor, where the density at any point is a (descending) function of the distance to the *k*th nearest data point. A decreasing function is called on some interval if for any values of the argument from this interval a larger value of the argument corresponds to a smaller value of the function. The inversion of the distance density to the *k*th nearest neighbor is taken to estimate the density. For each individual *i* the distances (in the criteria space) to the persons *j* in the archive and the general sample are calculated and stored in the list.

To calculate the value of fitness is used the value of the density of the location of individuals: for each individual and calculates the Cartesian distance from it to the rest of the individuals j in the archive and the set of individuals.

After ranking the list in ascending order, the *k*th element gives the desired distance to the person and is denoted σ_i^k. This σ_i^k denotes the distance from the individual *i* to the nearest *k*th neighbor. We use *k*, which is equal to the square root of the sample size. But it should be noted that quite often it is enough to use k =1, which leads to effective implementation.Then calculate the density value $D(i)$ for the individual *i*:

$$D(i) = \frac{1}{\sigma_i^k + 2}, \quad k = \sqrt{(N + N_A)},$$

where *N* this is the size of the population; N_a is the number of archives; *k* is approaching the nearest integer.

Two is added to the denominator to make sure its value is greater than zero and that $D(i) < 1$.

4) Finally, adding $D(i)$ to the initial value of fitness $R(i)$ individuals and gives it adaptability. Therefore, the final value of the fitness function $F(i)$ for the individual is defined as $F(i) = R(i) + D(i)$.

The execution time of the suitability determination procedure largely depends on the density estimate – $O(L^2 \log L)$, while the calculation of the values of *S* and $R - O(L^2)$, where $L = M + N$.

But in the end, basic SPEA-2 consists of the following stages:

At the entrance: *M*, *N*, *T*, *M* is the size of the original population. *N* is the size of the archive. *T* is the maximum size of generations.

Output: A^* – a nondominant set.

Step 1: *Initialization*: creating an initial set *P* and an empty archive – an external set $A_0 = \varnothing$, $t = 0$.

Step 2: *Calculation of fitness*: Calculation of values of fitness of each individual in P_t and $A_{\bar{t}}$.

Step 3: *Select the environment*: Copy all individuals from P_t and $A_{\bar{t}}$ to $A_{\bar{t}+1}$. If the size of $A_{\bar{t}+1}$ exceeds *N*, reduce $A_{\bar{t}+1}$ using the clipping operator, if $A_{\bar{t}+1}$ is less than *N*, fill $A_{\bar{t}+1}$ with the dominant individuals in P_t and $A_{\bar{t}}$.

Step 4: *Completion*: If t is greater than or equal to *T*, or another stop criterion is satisfied, then the set A^* is a set of solution vectors representing nondominant solutions in A_{t+1}. *End*.

Step 5: *Selection*: We use binary tournament selection with substitutions for A_{t+1} to fill the pool.

Step 6: *Variation*: Use crossover and mutation operators for the pool and set *P* as the result set. Increase the population counter ($t = t+1$) and go to *step 2*.

Let's give some comparison of the multiobjective genetic algorithms [1].

TABLE I. COMPARISON OF MULTIOBJECTIVE ALGORITHMS

Algorithm	Benefits	Disadvantages	Features
VEGA	Simplicity	Inability to produce Pareto-optimal solutions on convex surfaces	Selection operator
FFGA	Simplicity	Settings niche radius – σ_{share}	Rank-based definition or method of forming niches to determine fitness
NPGA	Simplicity with tournament selection in understanding, speed	Adjusting the radius of the niche, an additional parameter for tournament selection gives a more complex implementation	Tournament selection, no definition of fitness. Niche radius is a mechanism of diversity
NSGA	Fast convergence	Settings niche radius – σ_{share}	Adaptability is based on sorting nondominant solutions
SPEA	Well tested, clustering	Sophisticated clustering algorithm to maintain diversity. Not guaranteeing the preservation of the gran solutions	Adaptability is based on sorting nondominant solutions in an external set. Clustering to cut off the outer population.
SPEA-2	Preservation of boundary solutions	Difficulty in calculating fitness	Adaptability is based on Pareto strength (dominant solutions). The density is based on the method of *k*-nearest neighbors. There is elitism and external recruitment

IV. PROBLEM SOLUTION

To solve our problem A system for optimal selection of deep network parameters is proposed which contains the next components:

– chromosome formation component;

– component of multiobjective genetic algorithms;

– neural network learning component.

In Figure 3 an abbreviated scheme of the system for optimal selection of deep network parameters is shown. Of course, the genetic algorithms $A_1,\ldots, A_n$ mean the multiobjective genetic algorithms in the previous sections – VEGA, FFGA, NPGA, NSGA, SPEA, SPEA-2.

1) Chromose formation component. Here we form the first chromosome which has genes to store the next parameters: the number of layers, the number of neurons, the weigths.

2) Component of multiobjective genetic algorithms. Here we do calculations the first step of our system – to use genetic algotihms (GAs) to get chromoses for the next step. The algorithm is written a bit below.

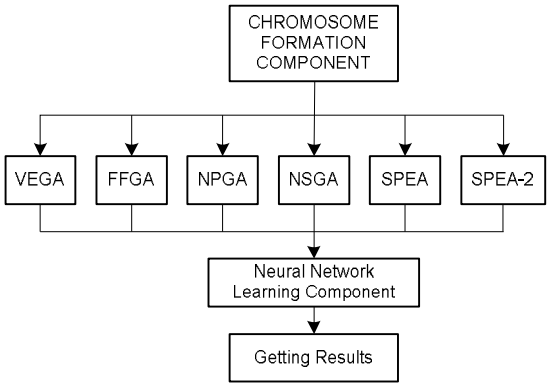

Fig. 3. Abbreviated block diagram of the multiobjective system for optimizing the configuration of neural networks

3) Neural network learning component. It is a component of the system that performs the stages of learning the neural network. The scheme of the component can be seen in Fig. 4.

4) The algorithm of choosing the optimal multiobjective GA. The whole process of genetic algorithms involved in the calculation subsystem depends on the algorithms included in it. All these

methods have their own specific system for determining the optimal parameters – i.e. the values of the genes of our chromosome).

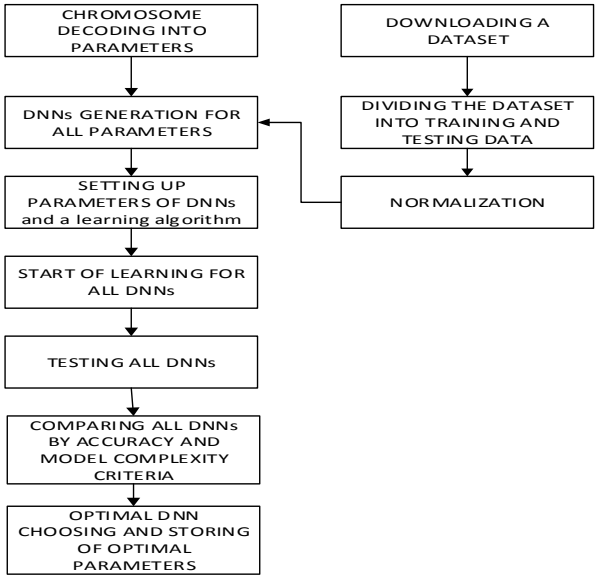

Fig. 4. Scheme of the learning component of neural networks

As a result, it is necessary to determine specifically the optimal multiobjective GA (genetic algorithm) for the selected type of problem. The algorithm begins with the filling of the chromosome, which structure we have already formed. Let R_k be the number of bits under the number of DNN layers, R_m be the number of bits under the number of neurons in each layer, and R_w be the number of bits responsible for the corresponding weights of neurons from the layers.

Algorithm steps

1. The initialization of the chromosome which stores the number of DNN layers, the number of neurons, the weights and the allocation of discharges for those genes.

2. Calculation of population size and number of generations (based on the fact that these parameters depend on chromosome size).

3. Initialization of chromosome formation, the number of which is equal to the size of the population and filling of genes of chromosomes with random bits using a random number generator.

4. Population copying in N multiobjective genetic algorithms.

5. For all genetic algorithms: Calculation of chromosome fitness. If the evolution is not complete, you need to go to the next step. Otherwise – step 8.

6. All genetic algorithms have the following: evolution with the help of GA operators (according to a specific algorithm). The operators are operators of selection, crossover, mutation.

7. All genetic algorithms have the following: obtaining a new generation the size of a population.

8. End of evolution and obtaining optimal parameters (number of layers, number of neurons for all layers, corresponding weights) in the form of chromosomes from all algorithms that go to the training of models from which the best of the models is selected. The learning process is shown in section in Fig. 4.

V. RESULTS

For testing our system the basic MNIST [13] was chosen.

TABLE II. RESULTS ON MNIST

Algorithm	Accuracy
VEGA	98.53%
FFGA	99.17%
NPGA	99.18%
NSGA	99.07%
SPEA	99.05%
SPEA2	99.45%

At the same time we have the next optimal parameters of our DNN model (Fig. 5).

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_27 (Dense)             (None, 364)               285740
_________________________________________________________________
batch_normalization_13 (Batc (None, 364)               1456
_________________________________________________________________
dropout_13 (Dropout)         (None, 364)               0
_________________________________________________________________
dense_28 (Dense)             (None, 52)                18980
_________________________________________________________________
batch_normalization_14 (Batc (None, 52)                208
_________________________________________________________________
dropout_14 (Dropout)         (None, 52)                0
_________________________________________________________________
dense_29 (Dense)             (None, 10)                530
=================================================================
Total params: 306,914
Trainable params: 306,082
Non-trainable params: 832
```

Fig. 5. Optimal parameters

If compare with TPOT Python package [14] then we have 0.05% accuracy with less model complexity.

The results of the system to reach

VI. CONCLUSIONS

The optimal choosing of paramaters for deep neural network was considered.

Overview of genetic algorithms was given.

The system to solve the problem of choosing optimal parameters was designed.

The effectiveness of the proposed system is confirmed by results.

REFERENCES

[1] Ghosh Ashish & Dehuri Satchidananda, "Evolutionary Algorithms for Multi-Criterion Optimization: A Survey," *International Journal of Computing & Information Sciences*, 2, 2004.

[2] C. A. C. Coello, "Evolutionary multi-objective optimization: a historical view of the field," *Comput. Intell. Mag. IEEE* 1 (1), 28–36, 2006. https://doi.org/10.1109/MCI.2006.1597059

[3] K. Deb, "Multi-Objective Optimization Using Evolutionary Algorithms," vol. 16, *John Wiley & Sons*, 2001.

[4] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, "Multi-objective evolutionary algorithms: a survey of the state of the art," *Swarm Evol. Comput*, 1 (1), 32–49, 2011. https://doi.org/10.1016/j.swevo.2011.03.001.

[5] B. Li, J. Li, K. Tang, and X. Yao, "Many-objective evolutionary algorithms: a survey," *ACM Comput. Surv.* 48 (1), pp. 1–35, 2015. https://doi.org/10.1145/2792984

[6] H. Ishibuchi, N. Tsukamoto, and Y. Nojima, "Evolutionary many-objective optimization: a short review," *Proceedings of the IEEE Congress on Evolutionary Computation*, 2008, pp. 2419–2426. https://doi.org/10.1109/CEC.2008.4631121

[7] M. Farina, and P. Amato, "A fuzzy definition of "optimality" for many-criteria optimization problems," *IEEE Trans. Syst. Man Cybern*, *Part A: Syst. Hum.*, 34 (3), 2004, pp. 315–326. https://doi.org/10.1007/s40747-019-0113-4

[8] Mario Köppen, Raul Vicente-Garcia, and Bertram Nickolay, "Fuzzy-Pareto-dominance and its application in evolutionary multi-objective optimization," in: *Proceedings of the Evolutionary Multi-criterion Optimization*, Springer, 2005, pp. 399–412. https://doi.org/10.1007/978-3-540-31880-4_28

[9] S. Yang, M. Li, X. Liu, and J. Zheng, "A grid-based evolutionary algorithm for many-objective optimization," *IEEE Trans. Evol. Comput.* 17 (5), 2013, pp. 721–736. https://doi.org/10.1109/TEVC.2012.2227145.

[10] R. Wang, R. C. Purshouse, and P. J. Fleming, "Preference-inspired coevolutionary algorithm for many-objective optimization," *IEEE Trans. Evol. Comput.* 17 (4), 2013, pp. 474–494. https://doi.org/10.1109/TEVC.2012.2204264.

[11] M. Li, S. Yang, and X. Liu, "Shift-based density estimation for Pareto-based algorithm in many-objective optimization," *IEEE Trans. Evol. Comput.*, 18 (3), 2014, pp. 348–365. https://doi.org/10.1109/TEVC.2013.2262178

[12] K. Tan, T. Lee, & E. Khor, "Evolutionary Algorithms for Multi-Objective Optimization: Performance Assessments and Comparisons," *Artificial Intelligence Review* 17, pp. 251–290, 2002. https://doi.org/10.1023/A:1015516501242

[13] kaggle.com/c/mnist-classification

[14] https://github.com/EpistasisLab/tpot

Received Oktober 21, 2020

Chumachenko Olena. orcid.org/0000-0003-3006-7460**.** Doctor of Engineering Science. Professor.
Technical Cybernetic Department, National Technical University of Ukraine "Ihor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine.
Education: Georgian Politechnic Institute, Tbilisi, Georgia, (1980).
Research area: system analysis, artificial neural networks.
Publications: more than 80 papers.
E-mail: chumachenko@tk.kpi.ua

Mykola Liubachenko. Bachelor.
Technical Cybernetic Department, National Technical University of Ukraine "Ihor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine.
Education: National Technical University of Ukraine "Ihor Sikorsky Kyiv Polytechnic Institute", Kyiv, (2019).
Research area: artificial neural networks.
Publications: 1.
E-mail: ejie1997@gmail.com

О. І. Чумаченко, М. О. Любаченко. Оптимальний вибір генетичного алгоритму для налаштування глибоких нейронних мереж

Розглянуто проблему налаштування глибоких нейронних мереж з використанням генетичних алгоритмів. Зазначено проблему структурно-параметричного синтезу при налаштування нейронних мереж. Основна мета дослідження – налаштувати глибоку нейронну мережу оптимально для розв'язку нагальних проблемі. Тип задачі для якої буде налаштовуватися глибока нейронна мережа – задача класифікації. Дано класифікацію генетичних алгоритмів, які використовуються як базис для налаштування параметрів глибоких нейронних мереж. Запропоновано систему оптимального налаштування параметрів глибоких нейронних мереж, до складу якої входить двоступеневий алгоритм. На першому кроці роботи алгоритму відбувається вибір багатокритеріального генетичного алгоритму із множини можливих (генетичний алгоритм векторної оцінки, генетичний алгоритм Фонсеки та Флемінга, генетичний алгоритм Парето-апроксимації з нішеванням,

генетичний алгоритм сортування без домінування, генетичний алгоритм сили Парето, генетичний алгоритм сили Парето-2), який найкращим чином підходить до заданої навчальної вибірки. На другому кроці розв'язується задача структурно-параметричного синтезу нейронної мережі за критеріями точності та складності. В результаті навчання знаходяться значення параметрів нейронної мережі такі як: кількість шарів, кількість нейронів в кожному шарі, значення вагових коефіцієнтів. Проведено моделювання запропонованої системи. Представлено результати моделювання, порівняння результатів з аналогічними програмними пакетами. Отримані результаті показують про можливості широкого використання.
Ключові слова: багатокритеріальний генетичний алгоритм; нейронна мережа глибокого навчання; структурно-параметричний синтез; оптимальний вибір.

Чумаченко Олена Іллівна. orcid.org/0000-0003-3006-7460. Доктор технічних наук. Професор.
Кафедра технічної кібернетики, Національний технічний університет України «Київський політехнічний інститут ім. Ігоря Сікорського», Київ, Україна.
Освіта: Грузинський політехнічний інститут, Тбілісі, Грузія, (1980).
Напрямок наукової діяльності: системний аналіз, штучні нейронні мережі.
Кількість публікацій: понад 80 наукових робіт.
E-mail:chumachecko@tk.kpi.ua

Любаченко Микола Олександрович. Бакалавр.
Кафедра технічної кібернетики, Національний технічний університет України «Київський політехнічний інститут ім. Ігоря Сікорського», Київ, Україна.
Освіта: Національний технічний університет України «Київський політехнічний інститут ім. Ігоря Сікорського», Київ, (2019).
Напрям наукової діяльності: штучні нейронні мережі.
Кількість публікацій :1.
E-mail: ejie1997@gmail.com

Е. И. Чумаченко, Н. А. Любаченко. Оптимальный выбор генетического алгоритма для настройки глубоких нейронных сетей

Рассмотрена проблема настройки глубоких нейронных сетей с использованием генетических алгоритмов. Указано проблему структурно-параметрического синтеза при настройке нейронных сетей. Основная цель исследования – настроить глубокую нейронную сеть оптимально для решения насущных проблем. Типзадачи для которой будет настраиваться глубокая нейронная сеть – задача классификации. Дана классификация генетических алгоритмов, которые используются как базис для настройки параметров глубоких нейронных сетей. Предложена система оптимальной настройки параметров глубоких нейронных сетей, в состав которой входит двухступенчатый алгоритм. На первом этапе работы алгоритма происходит выбор многокритериального генетического алгоритма из множества возможных (генетический алгоритм векторной оценки, генетический алгоритм Фонсеки и Флеминга, генетический алгоритм Парето-аппроксимации с нишерованием, генетический алгоритм сортировки без доминирования, генетический алгоритм силы Парето, генетический алгоритм силы Парето-2), который наилучшим образом подходит к заданной обучающей выборке. На втором шаге решается задача структурно-параметрического синтеза нейронной сети по критериям точности и сложности. В результате обучения находятся значения параметров нейронной сети такие как: количество слоев, количество нейронов в каждом слое, значения весовых коэффициентов. Проведено моделирование предложенной системы. Представлены результаты моделирования, сравнение результатов с аналогичными программными пакетами. Полученные результаты показывают о возможности широкого использования.
Ключевые слова: многокритериальный генетический алгоритм; нейронная сеть глубокого обучения; структурно-параметрический синтез; оптимальный выбор.

Чумаченко Елена Ильинична. orcid.org/0000-0003-3006-7460. Доктор технических наук. Профессор.
Кафедра технической кибернетики, Национальный технический университет Украины «Киевский политехнический институт им. Игоря Сикорского», Киев, Украина.
Образование: Грузинский политехнический институт, Тбилиси, Грузия, (1980).
Направление научной деятельности: системный анализ, искусственные нейронные сети.
Количество публикаций: более 80 научных работ.
E-mail:chumachecko@tk.kpi.ua

Любаченко Николай Александрович. Бакалавр.
Кафедра технической кибернетики, Национальный технический университет Украины «Киевский политехнический институт им. Игоря Сикорского», Киев, Украина.
Образование: Национальный технический университет Украины «Киевский политехнический институт им. Игоря Сикорского», Киев, (2019).
Направление научной деятельности: искусственные нейронные сети.
Количество публикаций: 1.
E-mail: ejie1997@gmail.com