REPUBLIC OF AZERBAIJAN

On the rights of the manuscript

ABSTRACT

of the dissertation for the degree of Doctor of Science

ARCHITECTURES AND MODELS OF NEURAL NETWORKS IMPLEMENTING METRIC METHODS OF RECOGNITION

Speciality: 3338.01. - System analysis, control and information processing

Field of science: Technics

Applicant: Geidarov Shahmali oglu Polad

Baku – 2021

The work was performed at Azerbaijan National Academy of Sciences, Institute of Control Systems, laboratory № 1.2 - Signal recognition methods and technical diagnostic systems

Scientific consultant: doctor of technical sciences, professor
Nusratov Oktay Qudrat oglu

Official opponents:

corresponding member of ANAS,
doctor of technical sciences, professor
Mammadova Masuma Huseyn gizi
doctor of technical sciences, professor
Ibragimov Bayram Ganimat Oglu
doctor of technical sciences, professor
Aliyev Elchin Rashid oglu
doctor of technical sciences, professor
Mammadov Javanshir Firudin oglu

Dissertation council ED 1.20 of Supreme Attestation Commission under the President of the Republic of Azerbaijan operating at Azerbaijan National Academy of Sciences Institute of Control Systems

Chairman of the Dissertation council:
academician of ANAS,
doctor of technical sciences, professor
_____ **Abbasov Ali Mamed oglu**

Scientific secretary of the Dissertation council:
doctor of technical sciences, professor
_____ **Musaeva Naila Fuad gizi**

Chairman of the scientific seminar:
doctor of technical sciences, dosent
_____ **Pashayev Farhad Heydar oglu**

GENERAL DESCRIPTION OF WORK

Relevance of the work. Neural networks are artificial intelligence systems that mimic the capabilities of a biological brain. Currently, neural networks are widely used in various tasks, but the most widespread use of neural networks is in image recognition problems. In these tasks, neural networks allow, based on the training input and output samples X, Y, to determine the correspondence of the input element to its image.

Neural networks have a number of advantages over other recognition methods. A neural network allows you to classify objects without the need to create and use strict recognition algorithms for a particular task. Currently, neural networks can achieve the best recognition results in comparison with existing other recognition algorithms. In technical implementation, neural networks also allow achieving a high speed of object recognition, due to a high level of parallelization of the recognition task performed by neurons. At the same time, the currently existing classical neural network architectures have a number of disadvantages.

There are different neural network architectures. In practical tasks, the multilayer perceptron in is most often used. A feature of the multilayer perceptron is the presence of more than one training layer (usually two or three). The procedure for creating these networks reveals the shortcomings of these neural networks, which are as follows:

1. When solving a particular problem, the network is selected based on the formulation of the problem and the available data for training. Supervised learning requires a “peer review” for each sample item. Sometimes obtaining such an estimate for a large data set is simply impossible.

2. After choosing the general structure, you need to experimentally select the network parameters. For networks like a perceptron, this will be the number of layers, the presence or absence of bypass connections, and the transfer functions of neurons. When

choosing the number of layers and neurons, one should proceed from the fact that the network's ability to generalize is the higher, the greater the total number of connections between neurons. On the other hand, the number of links is bounded from above by the number of records in the training data. Thus, at present there are no precise algorithms for the perceptron that clearly define the structure and topology of the neural network.

3. If in the process of training the neural network, the error on the training data continues to decrease, and the error on the test data increases, then the network has stopped generalizing and simply "remembers" the training data. This phenomenon is called network overfitting. In such cases, training is usually stopped. During the learning process, other problems may appear, such as paralysis, the network getting into a local minimum of the error surface. Thus, for classical networks it is impossible to predict in advance the manifestation of a particular problem, as well as to give unambiguous recommendations for their solution.

4. There is also no clear understanding of how certain settings of the weight and threshold values of neural network neurons affect the result of the neural network operation. In general, the internal content of the neural network, and in particular the perceptron network, is presented as a kind of black box that does not lend itself to clear understanding.

5. Even in the case of successful, at first glance, learning, the network does not always learn exactly what the creator wanted from it. Testing the quality of training a neural network should be carried out using examples that were not involved in its training. Moreover, the higher the quality of training, the greater the number of test cases. If neural network errors have a probability close to one billionth, then a billion test cases are needed to confirm this probability. It turns out that testing well-trained neural networks becomes a very difficult task.

A multilayer perceptron is a feedforward network, that is, a network that has no feedbacks. Neural network architectures that have feedbacks are called recurrent neural networks. The presence of

feedbacks allows you to memorize and reproduce whole sequences of reactions to one stimulus. Such features potentially provide many possibilities for modeling biological neural networks. But this class of networks also has its drawbacks:

1. Most of the possibilities for these networks are currently poorly understood due to the possibility of building various architectures and the complexity of their analysis.

2. The main disadvantage of these networks is the lack of stability, and in cases where it is achieved, the network becomes equivalent to a single-layer neural network due to which it is unable to solve linearly inseparable problems. As a result, the capacity of such networks is extremely small.

Here is another neural network architecture that has gained popularity in recent years - this is the architecture of deep learning neural networks, namely convolutional neural network. The architecture of convolutional neural networks consists in alternating convolutional layers and subsampling layers.

The peculiarity of these networks lies in the fact that the recognition process goes from layer to layer from simple elements-image details to complex images, concepts, etc. The advantage of the architecture of convolutional networks is that the network is able to distinguish and recognize objects in the image and, in addition, unlike the perceptron network, convolutional neural networks do not use a fully connected network architecture. Neural connections are selectively determined, which significantly reduces the number of customizable neural network weights. Despite all the advantages and popularity that these networks have received, nevertheless, the development of convolutional networks has a number of difficulties and uncertainties, and in particular the following:

1. The architecture of convolutional networks is very complex.

2. Too many variable network parameters, it is not clear for what task and computing power what settings are needed. So, the variable parameters include: the number of layers, the dimension of the convolution kernel for each of the layers, the number of kernels for each of the layers, the step of the kernel shift when processing the

layer, the need for subsampling layers, the degree of their reduction in dimension, the function to reduce the dimension (selection of the maximum average, etc.), the transfer function of neurons, the presence and parameters of the output fully connected neural network at the output of the convolutional one. All of these parameters significantly affect the result, but are chosen by researchers empirically.

Another neural network architecture is an impulse neural network (IMN) or spike neural network. This is the third generation of artificial neural networks (ANNs), which differs in that neurons exchange short pulses of the same amplitude (in biological neurons - about 100 mV). It is currently the most physiologically realistic INC model. A feature of these networks is that signals are transmitted to neurons in the form of impulses. The network receives a series of pulses at the inputs and transmits pulses at the output. At every moment, each neuron has some value (analogous electrical potential of biological neurons). And if this value exceeds the threshold, then the neuron sends a single impulse, after which its eigenvalue drops to a level below the average value by 2-30 ms. (an analogue of the rehabilitation process in biological neurons, the so-called refractory period). When leaving the state of equilibrium, the potential of the neuron begins to smoothly tend to the average value.

In frequency ANNs, a signal is used that takes on a value that depends on the frequency of generation of impulses by a certain group of neurons. (the weights of neurons, in fact, are a form of representation of this frequency). However, the average frequency of pulses in a sequence is a rather poor representation of information, since different types of stimulation can lead to the same average frequency of pulses. To get rid of this drawback in impulse ANNs, the following types of information presentation are used:

phase (time) - information about the signal is set by the exact (or within a certain window) position of the pulses in time (relative to some general reference rhythm of the brain);

synchronous positional - information about the signal is set by the synchronous activity of various groups of neurons, and, as a

consequence, the synchronous (or within a certain window) appearance of impulses at certain network outputs (for example, the auditory receptors of the cochlea that respond to high and low frequencies are located in different zones);

time before the appearance of the first impulse - information about the signal is set by the time of the appearance of the first impulse at any output;

ordinal - information about the signal is set by the order of receiving pulses at the network outputs;

interval - information about the signal is set by the distance between the pulses received at the network outputs;

resonant - information about the signal is set by a dense sequence of pulses, leading to the occurrence of resonance (single pulses decay and do not make any contribution to the transmission of information).

The main disadvantage of impulse neural networks (IMNS) is the lack of perfect learning algorithms implemented for these networks. This is due to the fact that well-known learning algorithms (such as back propagation) require the need to differentiate communication signals, but the differentiation of pulse sets of signals is a very complex and not fully understood task.

The difficulties associated with impulse neural networks are also associated with the fact that it is not quite clear how the coding and decoding of impulses in biological systems occurs, from where the idea of impulse signal transmission in neural networks was actually taken. The invented coding methods given above are not universal and have their drawbacks.

Another disadvantage of the IMS is the complexity of the implementation of impulse neural networks, both in software and hardware implementation. In software implementation, it is not very practical to implement signal transmission between neurons in the form of a beam of impulses. This leads to too much slowdown of the neural network. In a hardware environment, the implementation of impulse neural networks requires the creation and invention of new

types of neurons, which requires the creation of complex electronic hardware systems for one neuron.

Separately, it is also necessary to say about the learning algorithms for neural networks, and in particular the backpropagation method. The backpropagation algorithm computes the gradient vector of the error surface. This vector indicates the direction of the shortest descent along the surface from a given point, so if we move along it, the error will decrease. A sequence of such steps (slowing down as it approaches the bottom) will eventually lead to a minimum of one type or another.

Algorithms for training neural networks also have their own problematic sides, which in practice complicate the work of neural network developers.

A certain difficulty here is the question of how to take the length of the steps. With a large step length, the convergence will be faster, but there is a danger of jumping over the solution or go in the wrong direction. A classic example of such a phenomenon when training a neural network is when an algorithm moves very slowly along a narrow ravine with steep slopes, jumping from one side to the other. On the contrary, a small step will be in the right direction, but it will take a lot of iterations and the learning process will be too slow.

Despite the many successful uses of backpropagation, backpropagation is not a universal solution. The most troublesome thing is the indefinitely long learning process. In complex tasks, the network may take days or even weeks to train, or it may not learn at all. The reason may be one of the following. In the process of training the network, the values of the weights can become very large as a result of the correction. This can cause all or most of the neurons to function at very high values. This phenomenon is called network paralysis. This is usually avoided by decreasing the step size, but it increases the training time. Various heuristics have been used to prevent or recover from paralysis, but so far they can only be considered experimental.

The problem of retraining the network should also be noted. This phenomenon is more likely the result of an erroneous design of its topology or an incorrect choice of the criterion for stopping learning. During retraining, the network's ability to generalize information is lost. The entire set of images provided for training will be learned by the network, but any other images, even very similar ones, may be incorrectly recognized. In this way, we are minimizing the wrong error that really needs to be minimized - the error that can be expected from the network when completely new observations are fed to it.

The described problems with local minima and the choice of the network size lead to the fact that in practical work with neural networks, as a rule, one has to experiment with a large number of different networks. Sometimes training each of them several times and comparing the results.

The disadvantages of neural networks and learning algorithms listed above can be generalized and the following disadvantages can be added:

1. The complexity of choosing the structure of the architecture of the neural network and its parameters (the number of neurons, connections, layers). These parameters of the neural network are selected either empirically, or using approximate expressions, or have a very complex multilayer form, such as for deep learning networks.

2. The complexity of training a neural network, with a large number of recognizable patterns. Moreover, the greater the number of recognized patterns, the more difficult the neural networks are trained, and the worse, as a rule, the learning result. This leads to a limitation of the possible number of patterns (N) used in recognition problems using neural networks.

3. In the trained models, the values of the weights are determined on the basis of learning algorithms, the implementation of which requires a long execution and a sufficient volume of the trained sample. At the same time, the training results are not constant and depend not only on the set of the training sample and the

conditions of the training algorithms, but also on the parameters of the network structure.

4. Another difficulty is the problem of expanding the already trained neural network and adding new images to it. Adding new images to an already trained network requires performing the process of retraining the network with complete or partial changes in the previous weight and threshold values, which also significantly complicates the introduction of changes into the already trained neural network.

In connection with all of the above, the creation of new architectures of neural networks, with new capabilities, is a very aktual task today. In this regard, in this work, propose new architectures of neural networks that could allow solving the indicated problems of typical architectures of neural networks.

The main purpose of the work. The main goal of the work is to create technologies, algorithms and network architectures that make it possible to simplify and speed up the procedure for creating and training artificial neural networks, as well as making it possible to make multilayer perceptron networks transparent and understandable. For this purpose, in the dissertation work, the following goals and objectives were set and solved, consisting in the following points:

1. The aim of the work is to create new architectures of neural networks based on metric recognition methods, which allow analytically determining the network parameters: the number of neurons, layers and connections based on the initial conditions of the recognition problem, such as the number of samples and patterns.
2. The goal is also to determine and analytically calculate the weight and threshold values of the feedforward neural network (multilayer perceptron). This goal is achieved by implementing a neural network based on existing recognition algorithms for metric recognition methods.
3. The work also aims to study the resulting neural network architectures, their capabilities and advantages.
4. Including studies of the possibilities of applying the proposed

neural networks to various types of tasks, as well as the use of these architectures in multitasking applications.

5. Another set goal is to study the possibility of using the proposed architectures of neural networks for fuzzy logic tasks.

6. The goal is also to determine the ranges of weight values, and general laws, conditions and rules determining the distribution of their values.

7. The goal of mathematical substantiation of the connection of the proposed neural networks with the architecture of a fully connected multilayer perceptron is also set and also the relationship of the calculated weights with the weights of the trained neural network using classical learning algorithms.

8. Another set goal is to study the possibilities and ways of representing various kinds of input objects (images, symbols, texts, photographs) for their representation in the proposed neural network.

9. The goal is also set: performing various experiments with the proposed neural networks, including performing comparative experiments on the capabilities of the proposed neural network architectures with the capabilities of well-known classical neural networks, such as a fully connected multilayer perceptron, based on MNIST numbers (60,000 training sample and 10,000 control sample) , in order to analyze the possibilities and advantages of the proposed technologies with the well-known classical neural network technologies.

10. Another goal is, based on the proposed technologies, to implement the possibilities of determining the values of the weights of the neural network without calculating and training the values of the weights. The goal is to realize and describe such a possibility using physical expressions of the electrostatic field.

Applied methods. The work uses metric recognition methods, the theory of neural network technologies, the theory of fuzzy logic, learning algorithms for neural networks, image processing methods, mathematical modeling and programming, the theory of the foundations of electrostatics.

The main provisions to be defended.

- a solution that allows to immediately analytically calculate and determine the values of weights and thresholds of the multilayer perceptron neural network;
- a solution that allows to immediately analytically determine the structure of the neural network of a multilayer perceptron, based on the initial conditions of the recognition problem;
- estimation of the time of neural network creation and analytical calculation of the values of all weights;
- a solution that allows you to get a transparent neural network of a multilayer perceptron;
- a solution to reduce the likelihood of the neural network getting into local minima;
- solutions allowing to reduce the volume of the training base for training a multilayer perceptron;
- a solution that allows to optimize the structure of the neural network, minimizing the number of neurons and neural network connections;
- solutions that allow adding new images and tasks to the neural network without changing the previous weights and thresholds.
- performed theoretical and practical proofs, showing the acceleration of the procedure for creating and training a multilayer perceptron neural network based on the architectures of the proposed feedforward neural network architectures;
- evaluation of the training time and the obtained performance of the proposed neural network with classical training schemes for multilayer perceptron neural networks.
- a solution that allows instantly determine the values of the weights of the multilayer perceptron using the parameters of the electrostatic field;
- a software module created in the Builder C++ software environment that implements the proposed architectures of neural networks, as well as their capabilities and also allows you to compare the capabilities of the resulting architectures with the capabilities of classical neural networks;

Scientific novelty. The scientific novelty of the proposed work lies in the following provisions:

1. A whole class of new neural network architectures is proposed.

2. Shown a direct connection between classical architectures of neural networks and metric recognition methods.

3. Exact analytical expressions are shown that determine the structure of the neural network, based on the initial conditions of the problem.

4. Analytical expressions are shown that calculate the weight and threshold values of neurons based on expressions of proximity measures.

5. It is shown that the proposed technology for creating neural networks makes it possible to quickly apply feedforward neural networks in multitasking applications.

6. The connection of the obtained networks with the classical multilayer perceptron is shown, as well as the possibility of training the obtained networks with classical learning algorithms.

7. It is shown that for the proposed architectures of neural networks, the process of creating and training a neural network is much faster in comparison with classical training schemes.

8. The possibilities of expanding the obtained networks with the addition of new patterns to the recognition problem are shown, without the need to change the previous parameters.

9. The proposed technology for creating computable feedforward neural networks allows training neural networks with a smaller number of training samples.

10. The proposed technology of computed neural networks allows to significantly reduce the probability of network paralysis or the neural network getting into a local minimum during the learning process.

11. It is shown that the value of the weights of the proposed architecture of the neural network can be determined almost instantly (without calculations and training) using the parameters of the electrostatic field.

The practical value of the work.

1. The proposed algorithms and architectures of neural networks make it possible to speed up the creation and training of classical neural networks. This is provided due to the possibility of analytical calculation of the number of neurons, layers and connections of the neural network, as well as the calculation of the values of weights and thresholds of neurons and connections of the neural network. These capabilities allow to quickly create a workable neural network. Experiments have shown that the process of computing a neural network is performed in a fraction of a second. The training process of the proposed neural networks allows to speed up the process to training by 40 percent or more faster than training classical neural networks.

2. The proposed neural networks also make it possible to easily expand the neural network with the addition of new recognizable patterns to the recognition problem without the need to change the previous weight and threshold values, which will be especially important for recognition problems using a large number of recognized patterns.

3. The proposed neural networks are transparent, which also allows to optimize the structure of the neural network without changing the performance of the neural network.

4. The proposed technologies of computational neural networks can also significantly reduce the likelihood of network paralysis and the network hitting a local minimum in the learning process.

5. The proposed neural networks make it possible to reduce the amount of the required training set.

6. The proposed neural networks make it possible to simplify and speed up the procedure for using one neural network in multitasking applications.

7. The proposed neural networks are transparent, which can make it possible to understand the recognition algorithm in a trained feedforward neural network.

Approbation of work. The main results of the dissertation work were tested at Republican and International conferences, such

as: Proceedings of the VII International Symposium "Интеллектуальные системы", INTELS-2006, Krasnodar, Russia; Proceedings of the International Conference PCI - 2006 "Проблемы Кибернетики и информатики", Baku, 2006; The second international conference “Problem of Cybernetics and informaties.”, Bakı, 2008; Materials of 1 republican scientific-practical conference on problems of electronic science, Baku, 2012; International Scientific and Practical Conference "Научно-практические исследования", Russia, Omsk, 2020; XXVI International Scientific and Practical Conference "Advances in Science and Technology", Russia, Moscow, 2020.

Application of work. The proposed technology has been applied and repeatedly tested for the problem of recognition of handwritten Arabic numerals based on MNIST (Modified National Institute of Standards and Technology), consisting of 60,000 digit images for the training set and 10,000 digit images for the control set. The application of this work was carried out in the programming environment Pyton and Builder C ++.

Published works. On the topic of the dissertation work, 30 works were published. Of these, 26 papers were published without co-authors, 21 papers were published in foreign reviewed scientific journals, 12 are included in the Scopus database, 15 are included in the Web of Science database, 5 articles were published by the Springer editors. The list of published works is given at the end of the abstract.

The name of the institution where the dissertation work was performed. The dissertation work was carried out at the Institute of Control Systems of the National Academy of Sciences of Azerbaijan.

The structure of the scientific dissertation. The dissertation work consists of the Introduction (32 pages - 52,415 characters), 8 chapters (1 chapter - 54 pages - 50834 characters, 2 chapter - 14 pages - 13922 characters, 3 chapter - 24 pages - 26234 characters, 4 chapter - 41 pages - 42945 characters, 5 chapter - 54 pages - 53398 characters, 6 chapter - 13 pages - 14401 characters, 7 chapter - 23

pages - 24870 characters, 8 chapter - 30 pages - 38516 characters), Final conclusions (3 pages - 4798 characters) and Appendix. The dissertation work contains 161 figures and 24 tables. The main volume of the dissertation work is 299 pages (322480 characters) of typewritten text, excluding spaces in the text.

THE CONTENT OF THE WORK

In conducting the dissertation, the relevance and purpose of the work is considered. The most significant and used classical neural network architectures are considered, as well as the strengths and weaknesses of these architectures. Based on the existing problematic aspects of neural networks, the goal of the work is formed.

Chapter I, provides a general explanation of metric recognition methods and a description of some algorithms that implement metric recognition methods, and also proposed neural network architectures that implement metric recognition methods [8, 15, 25]. Analytical expressions are given that determine the weight values of the first layer, as well as analytical expressions that determine the structure of the neural network: the number of neurons, layers, connections [8, 15]. An algorithm for selecting the minimum set of samples for a neural network is also proposed [8, 14]. There is also a general algorithm for the analytical calculation of all values of the weights in the table of weights [26, 27]. Ways to speed up this algorithm are proposed. The estimation of the computation time of all values of the neural network weights is performed.

Paragraf 1.1. it is said about the hypothesis of compactness, on the basis of the existence of which the metric recognition methods are based.

Paragraf 1.2. the description of the algorithm of the method of construction of samples (method of samples) is given.

Paragraf 1.3. the description of the algorithm of the nearest neighbor method is given.

Paragraf 1.4. the description of the algorithm of the potential function method is given.

Paragraf 1.5. examples of analytical expressions are given that determine the weight values of neurons in the first layer [15, 21, 26], for example, in the simplest case, this can be the expression:

$$w_i = \left(y_{etal1\,A} - y_{testi}\right)^2 - \left(y_{etal2\,B} - y_{testi}\right)^2, \qquad (1)$$

where $y_{etal1\,A}$ and $y_{etal2\,B}$ are the values of the curves *etal₁* and *etal₂* for the *AB* column in fig. 1.

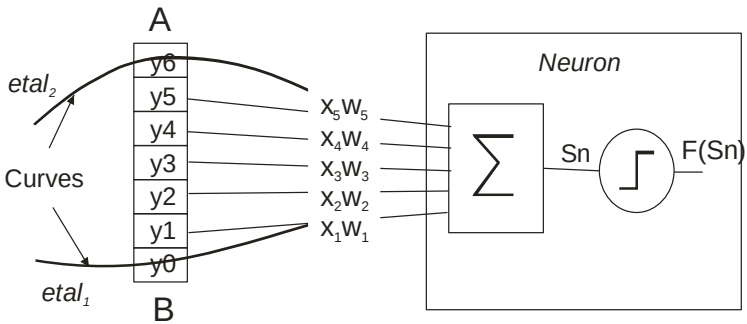


Fig. 1 Scheme of one neuron.

For example, if the value $y_{etal1\,A}$ = 6, $y_{etal2\,B}$ = 0 (fig. 1), then for different values y_{testi} we obtain the weights w_i for all sections between the two curves in the AB column:

$$\begin{aligned} w_1 &= (6-1)^2 - (0-1)^2 = 25 - 1 = 24, \\ w_2 &= (6-2)^2 - (0-2)^2 = 16 - 4 = 12, \\ w_3 &= (6-3)^2 - (0-3)^2 = 9 - 9 = 0, \\ w_4 &= (6-4)^2 - (0-4)^2 = 4 - 16 = -12, \\ w_5 &= (6-5)^2 - (0-5)^2 = 1 - 25 = -24, \end{aligned} \qquad (2)$$

The paragraf also provides an expression that determines the value of the state of the neuron of the first layer, (fig. 1):

$$Sn = \sum_{j=0}^{M} \sum_{i=0}^{K} x_{ij} w_{ij}, \qquad (3)$$

where x_{ij} the x value for the *i*-th row and *j*-th column w_{ij} - is the weight for the *i*-th row and *j*-th column; *M*, *K* - the number of columns and rows of the image (fig. 1).

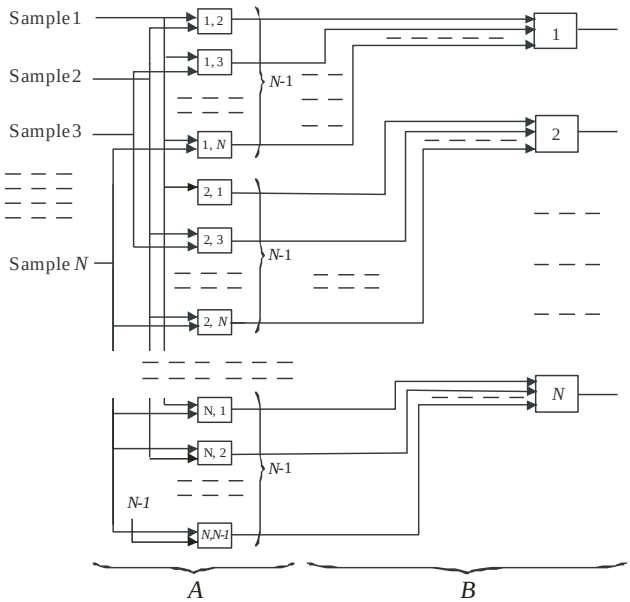


Fig. 2 Two-layer neural network with non-minimized architecture.

Paragraf 1.6. the architecture of a two-layer neural network (fig. 2), which implements the method of constructing samples (the method of samples) [8], is presented, and the activation functions of a neuron of the first (4) and second layers (5) are also considered:

$$f(Sn) = 1, \text{ if } Sn > 0; \qquad (4)$$
$$f(Sn) = 0, \text{ if } Sn < 0;$$

where $f(Sn)$ is the activation function of the neuron of the first layer (fig. 1, 2) and

$$f(Sn_i) = 1, \text{ if } Sn_i >= \alpha(N-1), \qquad (5)$$
$$f(Sn_i) = 0, \text{ if } Sn_i < \alpha(N-1),$$

where Sn_i is the value of the i-th neuron of the second layer, determined by the expression:

$$Sn_k^{(2)} = \sum_{j=1, j\neq k}^{N-1} \left(w_{k,j}^{(2)} f\left(Sn_{k,j}^{(1)} \right) \right), \qquad (6)$$

α - is the value of the weight w_i. In the simplest case, $\alpha = 1$. N is the number of samples. In this case, the number of neurons of the first

layer (n) of the neural network in fig. 2 is determined by the expression:

$$n = N(N-1) \tag{7}$$

and the number of neurons in the second layer is equal to the number of samples N.

Paragraf 1.7. the architecture of a two-layer neural network is presented in a condensed form fig. 3 [8]. This scheme of a neural network allows reducing the number of neurons (n) of the first layer by 2 times (9), based on the fact that neurons performing separation of identical pairs of samples are excluded from the neural network, for example {"sample 1", "sample 2"} and {" sample 2 "," sample 1 "}. In this case, in the second layer, the outputs of the excluded neurons of the first layer are replaced with the inverted values of the previous neurons (fig. 3b). The function of the state of the neuron of the first layer [15] is determined by the expression:

$$Sn^{(2)}{}_k = \alpha\left(\sum_{i=1}^{k-1} \overline{f}(Sn^{(1)}{}_{i\_k}) + \sum_{i=k+1}^{N} f(Sn^{(1)}{}_{k\_i})\right) \tag{8}$$

$$n = C_N^2 = \frac{N!}{2!(N-2)!} = \frac{N(N-1)}{2} \tag{9}$$

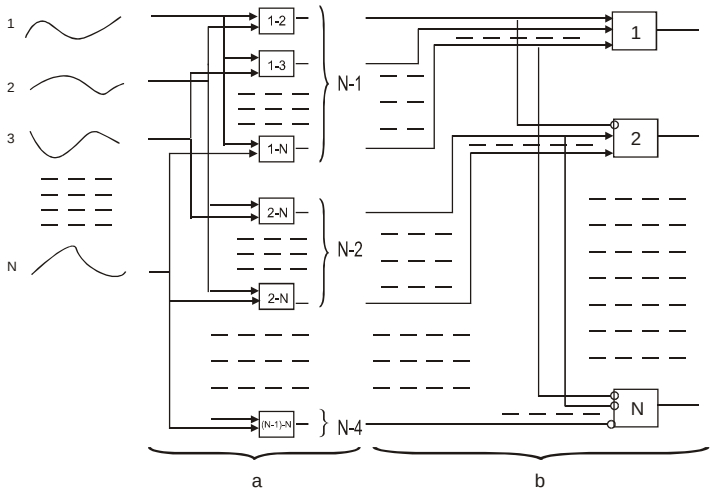


Fig. 3 Scheme of a two-layer neural network in compressed form for recognizing *N* curves.

Paragraf 1.8. a three-layer architecture that implements the nearest neighbor method is considered. For this, a third layer (fig. 4c) is added to the diagram in fig. 3, each neuron of which will be united by the sfmples of one image [21].

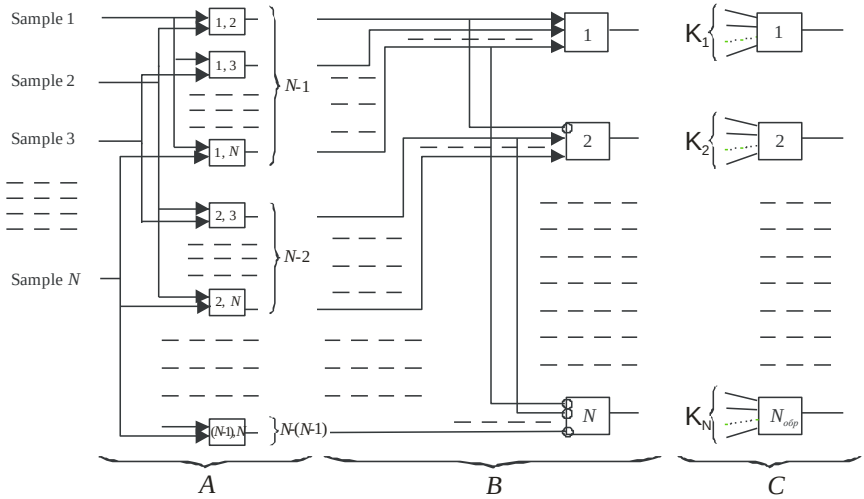


Fig. 4 Three-layer scheme with neural network architecture based on metric recognition methods.

The activation function of the neuron of the third layer is determined by the expression:

$$f^{(3)}(Sn^{(3)}_k) = 1, \text{ if } Sn^{(3)}_k > 0; \qquad (10)$$
$$f^{(3)}(Sn^{(3)}_k) = 0, \text{ if } Sn^{(3)}_k = 0;$$

where $Sn^{(3)}_k$- is the value of the *k-th* neuron of the third layer, which calculates by the expression

$$Sn_k^{(3)} = \sum_{j=1}^{K} \left(\alpha \times f\left(Sn_j^{(2)} \right) \right), \qquad (11)$$

where *j* - indicates the outputs of the second layer neurons belonging to the *k*-th image, *K* - the number of samples belonging to the *k*-th image.

Paragraf 1.9. the possibility of additional compression of the neural network is considered by excluding the neurons of the first layer [15], which perform the separation of the samples of one image. It is shown that the number of neurons in the first layer (*n*) will decrease and will be determined as follows:

$$n = n_1(N_{etal} - n_1) + n_2(N_{etal} - n_1 - n_2) + \ldots$$

$$+ n_j(N_{etal} - \sum_{i=1}^{j} n_i) + \ldots + n_{N-1}(N_{etal} - \sum_{i=1}^{N-1} n_i),$$

$$n = \sum_{j=1}^{N-1} n_j (N_{etal} - \sum_{i=1}^{j} n_i), \qquad (12)$$

where n_j, n_i , is the number of samples of the *j*-th and *i*-th image, N_{etal} - is the total number of all samplses, *N* - is the number of images.

Paragraf 1.10. possible examples of representation of samples are given, and examples of metric expressions that can be used to calculate tables of weights of neurons in the first layer [8, 15, 21].

For example, for black and white images of symbols, it is possible to use a geometric Euclidean measure of proximity. In this case, for each point of the image surface, the distance to each graphic image is determined. Based on the values of these data, a table of weights is compiled for each pair of images. In fig. 5 shows an example for one image point (x_t, y_t) in relation to symbols "7" and "2".

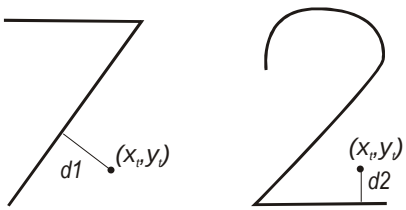


Fig. 5 Distances d1 and d2 for point (xt, yt).

Here, for the point (x_t, y_t) the smallest distances to each image will be equal to d_1 and d_2. Accordingly, the weight for this point will be determined as:

$$w_{ij} = d_1^2 - d_2^2 \ , \qquad (13)$$

where d_1^2 and d_2^2 - change $\left(y_{etal1\,j} - y_{testij}\right)^2$, $\left(y_{etal2\,i} - y_{testij}\right)^2$ and are determined by the expressions:

$$d_1^2 = \left(x_1 - x_t\right)^2 + \left(y_1 - y_t\right)^2, \tag{14}$$
$$d_2^2 = \left(x_2 - x_t\right)^2 + \left(y_2 - y_t\right)^2,$$

where (x_1, y_1) and (x_2, y_2) are the closest points of the sample images to the point (x_t, y_t) (Fig. 5).

Paragraf 1.11. in order to reduce the number of calculated weight tables, a zero layer is added to the neural network (fig. 6). Here, the activation function for a layer zero neuron is defined as:

$$f\left(Sn_{0,k}\right) = Sn_{0,k}, \tag{15}$$

where $Sn_{0,k}$ - is the value of the k neuron of the zero layer, determined by the expression:

$$Sn_{0,k} = \sum_{i=1}^{M}\sum_{j=1}^{K} x_{ij}\, w_{ij,k}, \tag{16}$$

where $w_{ij,k}$ - is the value of the weight of the i-th column and j-th row of the k-th table of weights, determined by the characteristic of the proximity measure used in the method.

In this case, weight tables are determined only for each neuron in the zero layer (Fig. 7). Since the number of neurons in the zero layer is equal to the number of samples, the number of calculated weight tables will also decrease from the values determined in expressions (7), (9), (12) to the number of N_{etal} samples. The value of the weights for the tables of the zero layer will be determined based on one samples, for example, in the simplest case, by the expression:

$$w_i = \left(y_{etal} - y_{\ i}\right)^2, \tag{17}$$

or by the expression:

$$w_{i,j}^{(0)} = d_1^2 = \left(x_i - x_t\right)^2 + \left(y_j - y_t\right)^2. \tag{18}$$

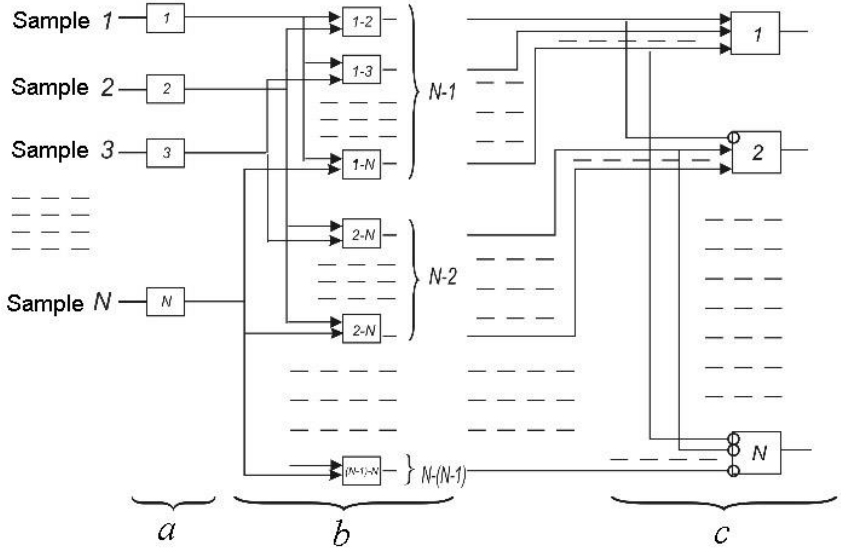


Fig. 6 Adding a zero layer: (a) - zero layer, (b) - first layer, (c) - second layer.

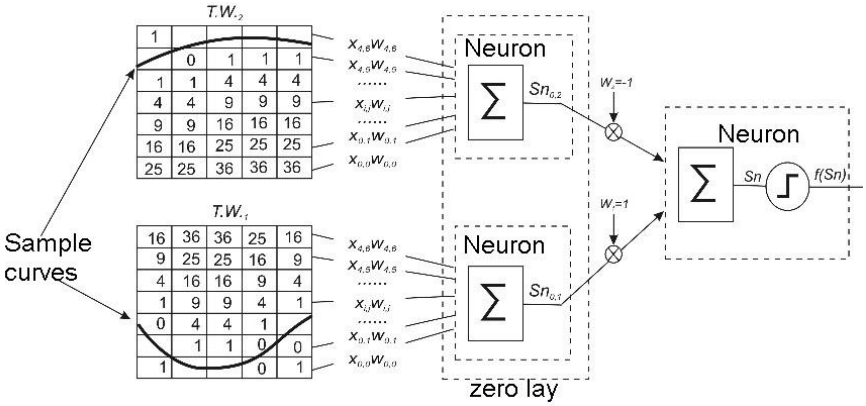


Fig. 7 Dividing a neuron of the first layer into two neurons of the zero layer.

Paragraf 1.12. an algorithm is proposed for selecting the minimum set of samples [15] for the proposed neural networks. If it

is not possible to select the samples, then to select the samples for a given task, you can use the existing algorithms for the training selection of samples. This paragraf also proposes an algorithm for the selection of samples, the work of which is as follows. The elements of the test set (Fig. 8) for the given image are assigned to the database one by one as a sample. After each assignment, full recognition is performed for all other elements of the sample, the results of which are recorded in the form of a code string consisting of zeros and ones, where 1 corresponds to an identified element, 0 - to an unidentified one (Tab. 1). If the number of units in a line corresponds to the required recognition reliability, then the current test element is selected as a sample for this pattern, and the algorithm ends there. If, after a similar procedure, the required sample is not determined, then the code line with the largest number of units in the line is selected, which is placed in the first line of the training table (Tab. 1), and the element corresponding to this line is placed in the samples data base. Further in this line, the numbers of the columns for which the values are equal to zero are determined, which corresponds to the numbers of the sample elements for which the recognition was not successful. In the future, the procedure is repeated in a similar way, but only now for these unidentified elements. Each of these elements is in turn added to the samples database, the code string is calculated, which is added to the training table (Tab. 1). The resulting training table becomes the object of research to determine the smallest set of rows, the bitwise logical addition of which gives the largest number of units in the final code line (Tab. 1), the value of which must not be less than the minimum required value. As a result of the operation of this algorithm, we obtain the minimum set of samples, the use of which provides the required level of reliability for this sample. It should be noted that this algorithm can be used for the sample method regardless of what set of features or what characteristic of the proximity measure is used in the method.

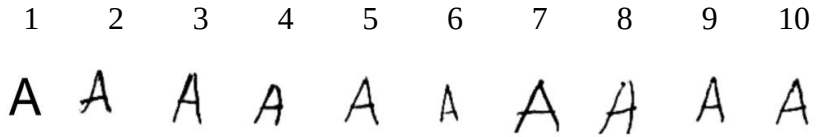


Fig. 8 A sample of ten (*n* = 10) hand-printed characters "*A*".

Tab. 1 Teaching table for character "*A*"

characters added to the base											
A – prn.	**1**	**0**	**1**	**0**	**1**	**1**	**1**	**0**	**1**	**1**	7
A-2	1	1	1	0	1	0	1	1	1	0	7
A-4	**1**	**1**	**0**	**1**	**1**	**1**	**0**	**1**	**1**	**0**	7
A-8	1	0	1	1	1	0	0	1	1	1	7
A–prn., A-4	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	10

Paragrafs 1.13.-1.16 provide examples demonstrating the calculation of weight tables and, based on the given schemes, examples of creating a neural network are given, as well as an example of implementing a neural network for a base of hand-printed characters and numbers.

In fig. 9 shows an example of calculating the values of a fragment of a weight table for two sample curves. Expression (1) was used as an expression for the analytical calculation of the values of the weights.

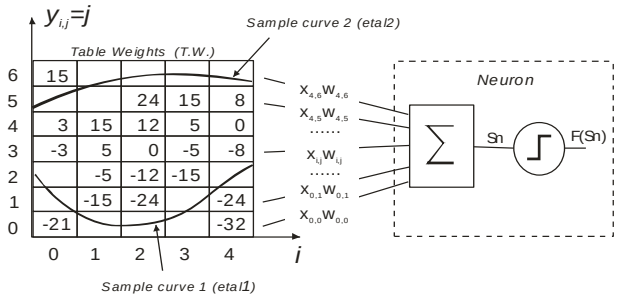


Fig. 9 Table of weights for two curves.

To determine the value of the weights of the table of weights for the problem of recognizing graphic symbols, a formula can be used that determines the smallest geometric distance to the image (13), (14), (Fig. 10ab).

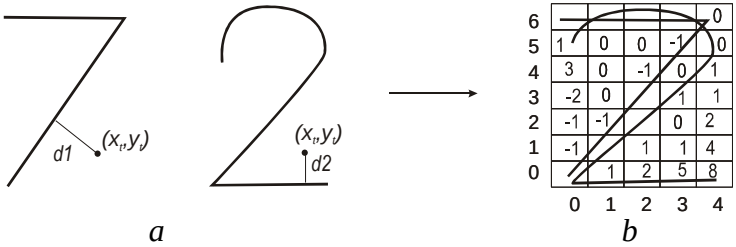


Fig. 10 Table of weights for two characters.

An example of creating the simplest neural network is shown in Fig. 12. In this example, a two-layer neural network is created for the problem of recognizing 3 patterns {7, 2, 4}. Tables of weights of dimension 5:7 for the selected image samples "7", "2" and "4" (Fig. 11), are determined on the basis of the expressions for the values of the weights (13), (14) for the neuron of the zero layer, and the values of the weights were determined by the geometric measure proximity (18).

Table 2 shows the result of the network operation as a result of testing the tested symbol "2" (Fig. 12), as well as the values and outputs of each neuron.

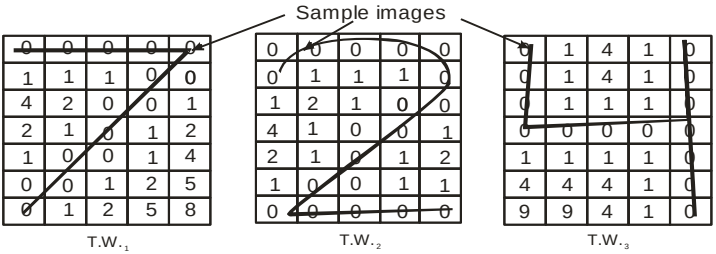


Fig. 11 Tables of weights of three characters.

Tab. 2 Neural network parameters in Fig. 12.

First layer			Second layer			*Yout*
k_m	Sn_{1,k_m}	$f(Sn_{1,k\_m})$	*k*	$Sn_{2,k}$	$f(Sn_{2,k})$	
1-2	20	0	1	$0+1=1$	0	-
1-3	-8	1	2	$\overline{0}+1=2$	1	{2}
2-3	-28	1	3	$\overline{1}+\overline{1}=0$	0	-

According to the results of the table, at the outputs of the neural network in fig. 12 the second network output corresponding to the image symbol {2} is activated.

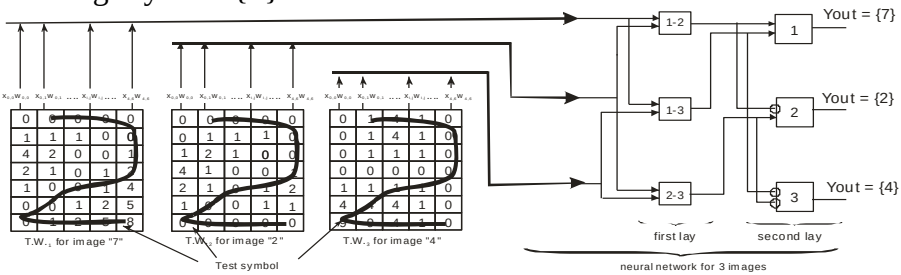


Fig. 12 Recognition of the test character "2" using a two-layer network for recognition of three patterns.

Paragraf 1.17. the description of the algorithm for calculating all values of the weight tables [26, 27], as well as the description of the operation of the program module implemented in the Builder C ++ environment based on this algorithm is given (Fig. 15).

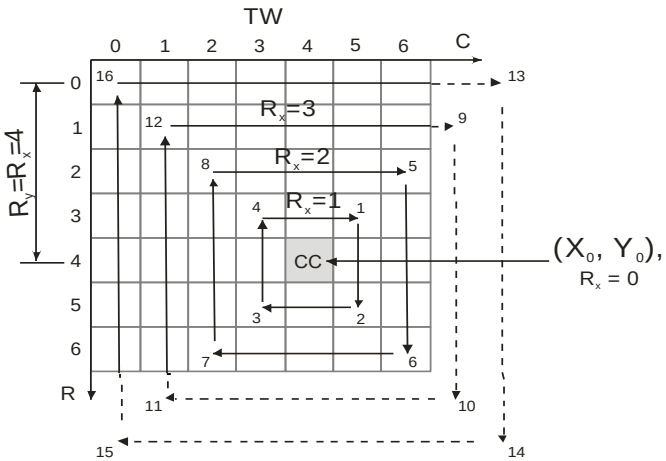


Fig. 13 Search by contours of the active cell relative to the current cell (CC) in the table of weights (TW).

The procedure for viewing the cells of the table of weights relative to the current cell (CC) with coordinates (4, 4)) is as follows (Fig. 13). The current cell is checked first. Next, a scan of the nearest cells of the table of weights located around the current cell is performed. The movement takes place along a closed contour, as shown in Fig. 13. The movement contour in the table is a square. The single contour scan procedure is implemented using four successive loops, where each loop loops through the cells on one side of the square. The first cycle - the right side of the square, performs a movement from top to bottom from point 1 to point 2 (Fig. 13); the second cycle is the lower side of the square, movement is performed from right to left from point 2 to point 3; the third cycle is the left side of the square, movement is performed from bottom to top from point 3 to point 4; the fourth cycle - the top side of the square, performs a movement from left to right from point 4 to point 1. In fig. 13 X_0, Y_0 - coordinates of the current cell, $X_0 = 4$, $Y_0 = 4$. R_x, Ry - contour radius along the X and Y axes relative to the current cell (Fig. 13). The initial values of R_x, R_y are taken equal to one. After viewing one contour, the R_x and R_y values are increased by one, after which

the cells are scanned along the closed contour, only now with new values $R_x = R_y$. For each cell of the table of weights meeted on the path of movement along a closed contour, the activity of this cell of the table is checked. Using the coordinates of the selected cell in the image X_1, Y_1, all pixels in this cell are scanned for the presence of darkened areas, the value of which is not higher than 50, which corresponds to darkened shades of a black and white image, where the tonality of a pixel is measured in the range from 0 to 255. When the condition of shading pixels in the table cells is met, the process is interrupted and the value 1 is returned, which corresponds to the activity of the cell. If, after scanning the entire cell, no active pixels are detected, then the value 0 is returned, which corresponds to the inactivity of this cell. Thus, all cells for the current cell are scanned and the nearest active cell for this sample is determined (examples in Fig. 14).

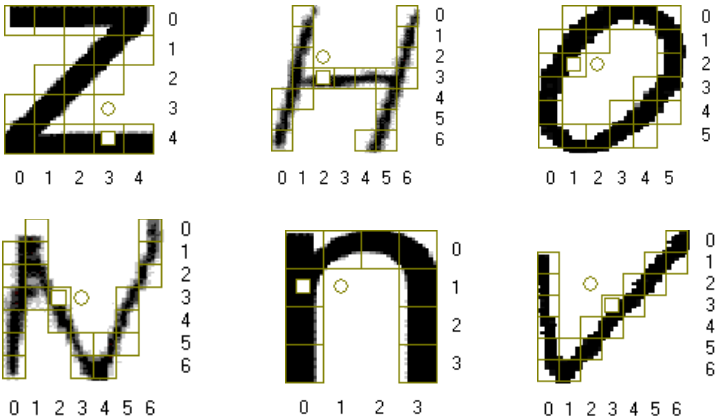


Fig. 14 Results of determining the nearest active cell (marked with a white square) relative to the current cell (marked with a white circle).

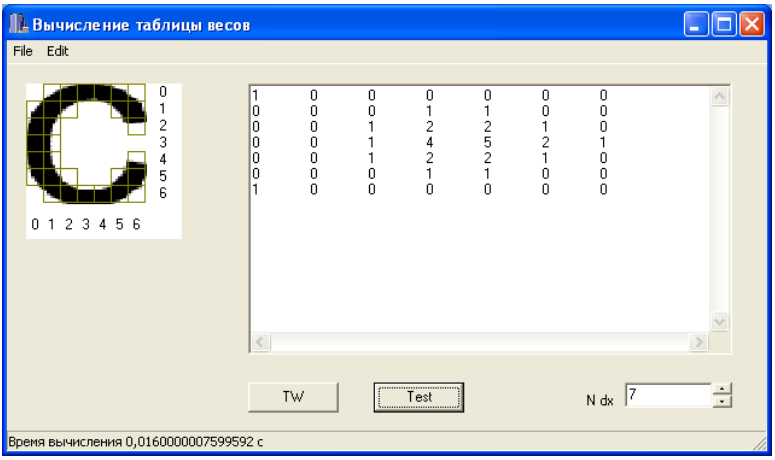


Fig. 15 Software module for calculating the table of weights of the zero layer neuron.

Paragraf 1.18. a method for accelerating the algorithm for calculating all values of the weight tables [26], which is also implemented in a software module, is described.

In the above paragraph, an algorithm is presented in which an enumeration of all the cells of the table of the selected dimension (Fig. 13). As it was shown above, for the current cell of the table, the search for active cells along the contours is performed by also enumerating all the cells of the table of weights. In this case, the time for calculating the weight tables can be significantly less if the proposed algorithm excludes the consideration of unnecessary contours and, accordingly, unnecessary cells of the weight tables. In particular, if the current cell itself with table coordinates (X0, Y0) is active, then there is no need to view all other cells.

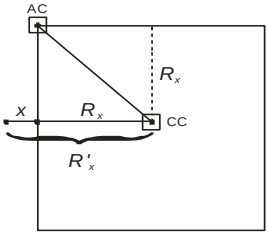


Fig. 16 Determination of x - the minimum sufficient number of viewed contours, after identifying the active cell AC.

In addition, if the minimum weight value is determined on some contour, then in this case there is also no need to consider all subsequent contours and, accordingly, table cells in these contours. In this case, it is necessary to determine the number *x* required to view the subsequent contours (Fig. 16). Since the maximum distance of the active cell (in Fig. 16 AC) at the current value of R_x from the center of the square (the current cell in Fig. 16 CC) will be at the vertices of the square and if the metric measure of the square of the Euclidean distance is used to calculate the weight value, then to reveal the value of x the following inequality must be solved:

$$(R_x + x)^2 < R_x^2 + R_x^2 \,, \tag{19}$$

$$x^2 + 2R_x x - R_x^2 < 0 \,. \tag{20}$$

To solve the inequality, we determine the roots of the quadratic equation:

$$x^2 + 2R_x x - R_x^2 = 0. \tag{21}$$

To do this, we determine the discriminant value *D* and the values x_1, x_2:

$$x_1 = \left(-2R_x\sqrt{2} - 2R_x\right)/2 = -R_x\left(\sqrt{2} + 1\right), \tag{22}$$

$$x_2 = \left(2R_x\sqrt{2} - 2R_x\right)/2 = R_x\left(\sqrt{2} - 1\right). \tag{23}$$

The solution to inequality (19) and (20) is the set of values

$$-R_x\left(\sqrt{2} + 1\right) < x < R_x\left(\sqrt{2} - 1\right), \tag{24}$$

and the maximum required value

$$x_{max} = R_x\left(\sqrt{2} - 1\right). \tag{25}$$

From here we determine the required maximum value of R'_x (Fig. 16), which can be no more than $R_x\sqrt{2}$, which follows from the expression:

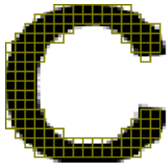
$$R'_x < x + R_x = R_x\left(\sqrt{2} - 1\right) + R_x = \\ = R_x\sqrt{2} \approx R_x \times 1{,}414. \tag{26}$$

Since the value $\sqrt{2} < 1{,}5 = 3/2$, to simplify the calculation, you can define the test condition as:

$$R'_x < 3R_x/2 \,. \tag{27}$$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	29	20	13	8	4	1	0	0	0	0	1	4	8	13	20	29
1	26	17	10	5	2	1	0	0	0	0	1	2	5	10	17	25
2	25	16	9	4	1	0	0	0	0	0	0	1	4	8	13	20
3	20	13	8	4	1	0	0	0	0	0	0	1	2	5	10	17
4	17	10	5	2	1	0	0	0	1	0	0	0	1	4	9	16
5	13	8	4	1	0	0	0	1	2	1	0	0	1	4	8	13
6	10	5	2	1	0	0	1	2	4	1	0	0	1	2	5	10
7	9	4	1	0	0	0	1	4	4	1	0	0	0	1	4	8
8	8	4	1	0	0	0	1	4	4	1	0	0	0	1	2	5
9	5	2	1	0	0	0	1	1	1	1	1	0	0	0	1	4
10	4	1	0	0	0	0	0	0	0	0	0	0	0	0	1	2
11	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
12	1	0	0	0	1	1	1	1	1	1	1	1	0	0	0	1
13	1	0	0	0	1	4	4	4	4	4	4	1	0	0	0	1
14	1	0	0	1	2	5	9	9	9	9	5	2	1	0	0	0
15	0	0	0	1	4	8	13	16	16	13	8	5	2	1	0	0

a



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	8	5	2	1	1	0	0	0	0	0	0	0	0	1	2	4
1	5	2	1	0	0	0	0	0	0	0	0	0	0	0	1	1
2	2	1	0	0	0	0	0	1	1	1	1	0	0	0	0	0
3	1	0	0	0	0	0	1	2	4	4	2	1	0	0	0	0
4	0	0	0	0	1	1	2	5	8	8	5	2	1	0	0	0
5	0	0	0	0	1	4	5	8	13	13	8	5	2	1	0	1
6	0	0	0	1	2	5	10	13	18	18	13	8	5	2	1	2
7	0	0	0	1	4	8	13	20	25	25	18	13	8	5	4	5
8	0	0	0	1	4	9	16	25	34	32	25	18	13	10	9	9
9	0	0	0	1	4	8	13	20	25	25	18	13	8	5	4	4
10	0	0	0	1	2	5	10	13	16	16	13	8	5	2	1	1
11	0	0	0	0	1	4	5	8	9	9	8	5	2	1	0	0
12	0	0	0	0	1	1	2	4	4	4	4	2	1	0	0	0
13	1	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0
14	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
15	5	2	1	0	0	0	0	0	0	0	0	0	0	0	1	2

b

Fig. 17 Tables of weights for printed characters "A" (a) and "C" (b) with dimension 16:16 for layer zero neurons.

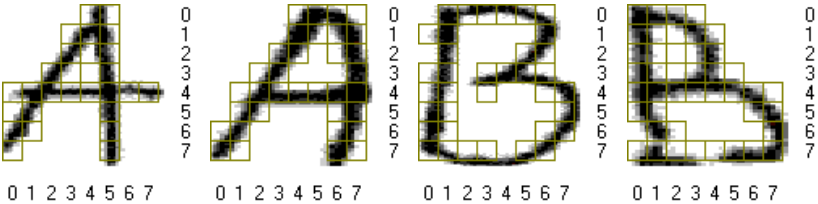
In fig. 17 shows a table of weights for the sample printable characters "*A*" (Fig. 17a) and "*C*" (Fig. 17b) with a dimension of 16 lines by 16 columns. These tables correspond to the tables of the weights of layer zero neurons. You can also see from the figure that the active table cells have a weight value equal to 0. The data in the table of weights was calculated using the metric measure of proximity - the squared Euclidean distance using the above algorithm.

In Paragraf 1.19, on the basis of the implemented program module, the computation time of the weight table is estimated [26]. The time for calculating the tables of weights T for all neurons in the zero layer is defined as $T = n \times t_{av}$ where t_{av} - is the average time for calculating one table of weights, n - is the number of neurons in the first layer. For example, if we are based on Fig. 17, we determine the average time for calculating the table of weights as t_{av} = (0.078 + 0.093) / 2 = 0.0855 s., then for the problem of recognizing alphabetic characters with the number of recognizable patterns equal to 30, in which 3 samples are used for each image (which in total is 90

samples), the computation time will be $T = 90 \times t_{\text{cp}} \approx 7$ s. for a neural network with a zero layer, and $T = 90 \times 89 \times t_{\text{cp}} = 6{,}23$ min. for the architecture of an extended neural network without a zero layer and $T = 90 \times 89 \times t_{\text{cp}}/2 \approx 3$ min. for a compressed neural network scheme [8, 15].

It should be noted that training a neural network with classical algorithms requires much more time, which is measured in hours, and sometimes days and weeks. Thus, the calculation of the value of the weights requires relatively little time, while an already working neural network is created, which can even be trained by classical algorithms.

Paragraf 1.20 provides an example [27] of calculating the values of weights using the implemented algorithm for calculating all values of the tables of weights for the problem of recognizing 6 patterns of handwritten letters {*A*, *B*, *E*, *M*, *N*, *Z*} with 8 samples shown in Fig. 18. In fig. 18 it can be seen that the images of the symbol "*A*" and "*B*" correspond to two samples, named as: A_1, A_2, B_1, B_2. For clarity and simplicity of the example, we will also use the scheme of a zero layer neural network with an extended scheme. For our problem, $N = 8$ (the number of samples). Based on the condition of the problem, the number of neurons in the zero layer is $n_0 = 8$, the number of neurons in the first layer is $n_1 = 8*7=56$, the number of neurons in the second layer is equal to the number of samples $n_2 = 8$, the number of neurons in the third layer is equal to the number of patternes $n_3 = 6$. The dimension of the table of weights is assumed to be 8:8, and thus the number of connections of one neuron in the zero layer will be equal to 64. In fig. 18 shows the tables of weights for each neuron of the zero layer, calculated as described above. Each neuron of the zero layer corresponds to one calculated table of weights in Fig. 18 according to their location in fig. 18. The functions of the states of the neurons of the zero layer (Fig. 20) are determined by multiplying the tables of weights with the binary matrix of the input recognizable element (Fig. 19ab). The functions of the states of the neurons of the first layer are given in table 3.



$\overline{W}^{(0)}_{(A1)}$

10	5	2	1	0	0	1	4
8	4	1	0	0	0	1	4
4	2	1	0	0	0	1	4
1	1	0	0	1	0	1	1
0	0	0	0	0	0	0	0
0	0	1	1	1	0	1	1
0	0	1	4	1	0	1	4
0	1	2	4	1	0	1	4

$\overline{W}^{(0)}_{(A2)}$

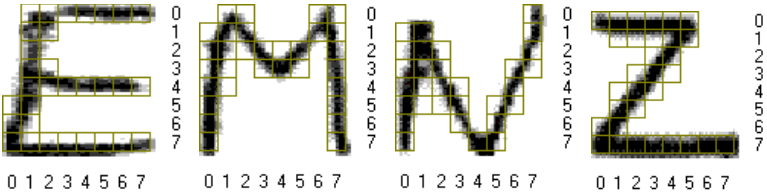
13	8	4	1	0	0	0	0
8	5	2	1	0	0	0	0
5	2	1	0	0	1	0	0
2	1	0	0	1	1	1	0
1	0	0	0	0	0	0	0
1	0	0	1	1	1	0	0
0	0	1	2	4	1	0	0
0	0	1	4	4	1	0	0

$\overline{W}^{(0)}_{(B1)}$

1	0	0	0	0	0	0	1
0	0	1	1	1	1	0	1
1	0	1	1	1	0	0	1
0	0	1	0	0	0	0	1
0	0	1	0	1	1	0	0
0	0	1	1	2	2	1	0
0	0	1	1	2	1	0	0
0	0	0	0	1	0	0	1

$\overline{W}^{(0)}_{(B2)}$

0	0	0	0	1	2	5	10
0	0	0	0	0	1	4	8
0	0	1	0	0	1	2	4
0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0
0	0	1	1	1	1	0	0
1	0	0	1	1	1	1	0
0	0	0	0	0	0	0	0



$\overline{W}^{(0)}_{(E)}$

1	0	0	0	0	0	0	0
1	0	0	1	1	1	1	1
1	0	1	2	4	4	4	4
1	0	0	1	1	1	1	1
1	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1
0	0	1	1	1	1	1	1
0	0	0	0	0	0	0	0

$\overline{W}^{(0)}_{(M)}$

1	0	0	1	4	1	0	0
0	0	0	1	1	1	0	0
0	0	0	0	0	0	1	0
0	0	1	0	0	0	1	0
0	0	1	1	1	1	1	0
0	1	2	4	4	4	1	0
0	1	4	8	9	4	1	0
0	1	4	9	9	4	1	0

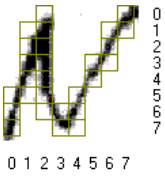
$\overline{W}^{(0)}_{(N)}$

1	1	2	5	8	4	1	0
0	0	1	2	5	4	1	0
0	0	0	1	4	2	1	0
0	0	0	1	2	1	0	0
0	0	0	0	1	1	0	1
0	0	0	0	1	0	0	1
0	1	1	0	1	0	1	2
0	1	2	1	0	0	1	4

$\overline{W}^{(0)}_{(Z)}$

0	0	0	0	0	0	1	4
1	0	0	0	0	0	1	4
2	1	1	0	0	1	2	5
2	1	0	0	0	1	4	8
1	0	0	0	1	2	5	9
1	0	0	1	2	4	4	4
0	0	1	1	1	1	1	1
0	0	0	0	0	0	0	0

Fig. 18 Sample symbols {*A1, A2, B1, B2, E, M, N, Z*} with the dimension of the image splitting matrix 8: 8 and the corresponding tables of weights of layer zero neurons.



(a)

	0	1	2	3	4	5	6	7
0	0	0	1	0	0	0	0	1
1	0	1	1	0	0	0	1	1
2	0	1	1	0	0	0	1	0
3	0	1	1	0	0	1	1	0
4	0	1	1	0	0	1	0	0
5	1	1	1	0	1	0	0	0
6	1	0	1	1	1	0	0	0
7	1	0	0	1	1	0	0	0

(b)

Fig. 19 (*a*) The tested handwritten character "*N*", (*b*) The binary matrix of the tested character $\overline{X}$.

$$\bar{X}\bar{W}^{(0)}_{(A1)}$$

0	0	2	0	0	0	0	4
0	4	1	0	0	0	1	4
0	2	1	0	0	0	1	0
0	1	0	0	0	0	1	0
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	0	1	4	1	0	0	0
0	0	0	4	1	0	0	0

$$Sw_1^{(0)} = 35$$

$$\bar{X}\bar{W}^{(0)}_{(A2)}$$

0	0	4	0	0	0	0	0
0	5	2	0	0	0	0	0
0	2	1	0	0	0	0	0
0	1	0	0	0	1	1	0
0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0
0	0	1	2	4	0	0	0
0	0	0	4	4	0	0	0

$$Sw_2^{(0)} = 34$$

$$\bar{X}\bar{W}^{(0)}_{(B1)}$$

0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	1
0	0	1	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	0	0	1	0	0
0	0	1	0	2	0	0	0
0	0	1	1	2	0	0	0
0	0	0	0	1	0	0	0

$$Sw_3^{(0)} = 15$$

$$\bar{X}\bar{W}^{(0)}_{(B2)}$$

0	0	0	0	0	0	0	10
0	0	0	0	0	0	4	8
0	0	1	0	0	0	2	0
0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
1	0	0	1	1	0	0	0
0	0	0	0	0	0	0	0

$$Sw_4^{(0)} = 31$$

$$\bar{X}\bar{W}^{(0)}_{(E)}$$

0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1
0	0	1	0	0	0	4	0
0	0	0	0	0	1	1	0
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	0	1	1	1	0	0	0
0	0	0	0	0	0	0	0

$$Sw_5^{(0)} = 14$$

$$\bar{X}\bar{W}^{(0)}_{(M)}$$

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0
0	0	1	0	0	0	1	0
0	0	1	0	0	1	0	0
0	1	2	0	4	0	0	0
0	0	4	8	9	0	0	0
0	0	0	9	9	0	0	0

$$Sw_6^{(0)} = 51$$

$$\bar{X}\bar{W}^{(0)}_{(N)}$$

0	0	2	0	0	0	0	0
0	0	1	0	0	0	1	0
0	0	0	0	0	0	1	0
0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0
0	0	1	0	1	0	0	0
0	0	0	1	0	0	0	0

$$Sw_7^{(0)} = 11$$

$$\bar{X}\bar{W}^{(0)}_{(Z)}$$

0	0	0	0	0	0	0	4
0	0	0	0	0	0	1	4
0	1	1	0	0	0	2	0
0	1	0	0	0	1	4	0
0	0	0	0	0	2	0	0
1	0	0	0	2	0	0	0
0	0	1	1	1	0	0	0
0	0	0	0	0	0	0	0

$$Sw_8^{(0)} = 27$$

Fig. 20 Tables of values of multiplication of the binary matrix $\bar{X}$ of the tested (input) symbol with tables of weights of the neurons of the zero layer $\bar{W}^{(0)}$ and the values of the state functions $Sw_i^{(0)}$ of the neurons of the zero layer.

Tab. 3 Values of state functions $Sw_{i,j}^{(1)}$ neurons of the first layer.

				A1	A2	B1	B2	E	M	N	Z
			j	1	2	3	4	5	6	7	8
			$Sw_j^{(0)}$	35	34	15	31	14	51	11	27
	i	$Sw_i^{(0)}$		$Sw_{i,j}^{(1)} = Sw_i^{(0)} - Sw_j^{(0)}$							
A1	1	35			1	20	4	21	-16	24	8
A2	2	34		-1		19	3	20	-17	23	7
B1	3	15		-20	-19		-16	1	-36	4	-12
B2	4	31		-4	-3	16		17	-20	20	4
E	5	14		-21	-20	-1	-17		-37	3	-13
M	6	51		16	17	36	20	37		40	24
N	7	11		-24	-23	-4	-20	-3	-40		-16
Z	8	27		-8	-7	12	-4	13	-24	16	

Tab. 4 Values of outputs of neurons of the first, second and third layers.

			A1	A2	B1	B2	E	M	N	Z		Second layer		Third layer	
		j	1	2	3	4	5	6	7	8					
	i		$f\left(Sw_{i,j}^{(1)}\right)$									$Sw_k^{(2)}$	$f\left(Sw_k^{(2)}\right)$	$Sw_{\text{обр}}^{(3)}$	Y_{out}
A1	1			0	0	0	0	1	0	0		1	0	0+0=0	0
A2	2		1		0	0	0	1	0	0		2	0		
B1	3		1	1		1	0	1	0	1		5	0	0+0=0	0
B2	4		1	1	0		0	1	0	0		3	0		
E	5		1	1	1	1		1	0	1		6	0	0	0
M	6		0	0	0	0	0		0	0		0	0	0	0
N	**7**		**1**	**1**	**1**	**1**	**1**	**1**		**1**		**7**	**1**	**1**	**1**
Z	8		1	1	0	1	0	1	0			4	0	0	0

Table 4 it can be seen that already at this stage the 7-th output of the second layer, corresponding to the sample "*N*", is active. In the third layer of the neural network, the neurons of the same image are combined in one neuron. For this example, the third layer is necessary for the first and second images "*A*" and "*B*", since these images have two samples and their outputs are combined in the neurons of the third layer. The state $Sn_k^{(3)}$ and $f\left(Sn_k^{(3)}\right)$ activation function of the third layer neuron are determined by formulas (10), (11).

From table 4, you can see that as a result, for the input symbol in Fig. 19a, the 5-th output of the neural network, corresponding to the "*N*" pattern, is activated.

Paragraf 1.21. the conclusions for the first chapter are given.

Chapter II explores the possibilities of the proposed neural network schemes for possible use in multitasking implementations [14, 15]. The advantages of the proposed neural network architectures in multitasking applications are considered.

Paragraf 2.1. the feasibility of creating a single classifier [14] in multitasking using a three-layer neural network with a zero layer is considered.

Paragraf 2.2. the feasibility of creating a single classifier in multitasking using a four-layer [14] neural network with a zero layer is considered.

Paragraf 2.3. a diagram of the application of one neural network to different tasks is given for the case when the belonging of the input object to a particular task is known [15]. In fig. 21 shows a general scheme for the application of a neural network in multitasking use with variously defined classifiers, for the case if the parameters for each applied task, such as the number of samples used, the number of sample features used, the number of recognized patterns - will correspond to the parameters of the applied neural network: the number of neurons of zero layer, dimensions of weight tables, the number of outputs of the neural network [8, 15].

In this paragraf, schemes for the implementation of a neural network in multitasking are investigated for cases when the input initial parameters used in recognition problems, such as: the number of samples, the dimension of the table of weights, differ from problem to problem.

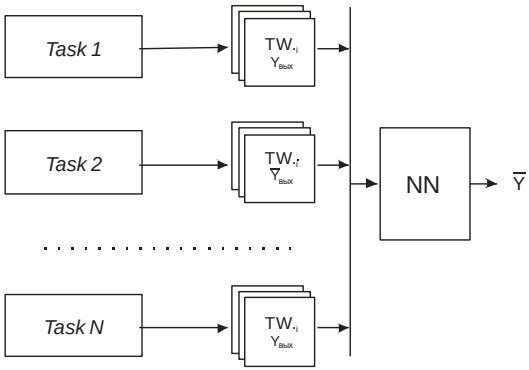


Fig. 21 Scheme of multitasking neural network application.

In this paragraf, we consider the cases of applying tasks to a neural network, in which the network parameters do not converge with the parameters of the applied problem. For example:

1. If for some problem, the dimension of the table of weights (m, k) is less than the dimension of the table of weights (M, K) of the neural network ($m < M$, $k < K$) (Fig. 22), then in this case the unused columns and the rows of the table can be reset to zero (Fig. 22), which, according to expression (28), will preserve the integrity and logic of the neuron of the zero layer:

$$Sn_0 = \sum_{i=1}^{M}\sum_{j=1}^{K} x_{ij}w_{ij} = \sum_{i=1}^{m}\sum_{j=1}^{k} x_{ij}w_{ij} + \sum_{i=m+1}^{M}\sum_{j=k+1}^{K} x_{ij}w_{ij} =$$
$$= \sum_{i=1}^{m}\sum_{j=1}^{k} x_{ij}w_{ij} + \sum_{i=m+1}^{M}\sum_{j=k+1}^{K} (x_{ij}\times 0) = \sum_{i=1}^{m}\sum_{j=1}^{k} x_{ij}w_{ij} \qquad (28)$$

where m, k - the number of columns and rows of the table of weights of the problem (in Fig. 22a m = 3, k = 4); M, K - the number of columns and rows of the table of weights of the neural network (in Fig. 22b M = 5, K = 7).

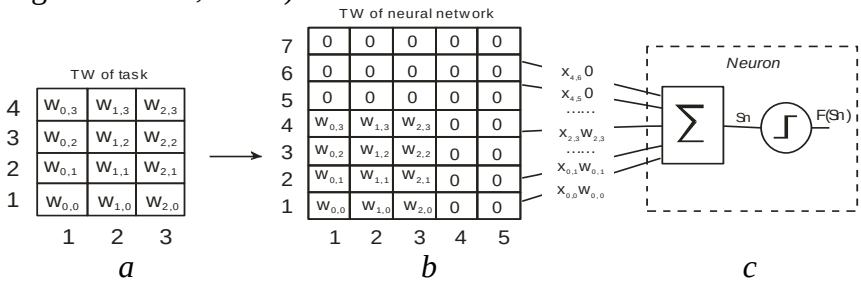


Fig. 22 Scheme of using the table of weights for a problem with a dimension less than the number of connections of a neuron.

2. If the number of samples for some task is less than the number of neurons in the zero layer, then in order to preserve the integrity of the network for this task, a table of weights of the "remote image" is created for non-effective neurons of the zero layer (*T.V.r.i.*), characterized by a significant distance from all other samples. In this case, the sample of the "remote image" is created on the basis of the condition under which the comparison between any

two samples, using the proximity characteristic used in the metric recognition method, was less than the identical comparison of any samples with the sample of the "remote image" [14]. To fulfill this condition, it is sufficient to satisfy condition (29), under which all values of the weights of the *T.W.r.i.* take one value (w_r) greater than the value of the maximum weight (w_{max}) determined from the set of all weight values (w_{ij}, k) in the set of real weight tables.

$$w_{ij,k} \le w_{\max} \le w_r \tag{29}$$

In this case, the value of wr can be defined as:

$$w_r = Lw_{\max}\ , \tag{30}$$

where $L \in [1, \infty]$ is the coefficient of remoteness of the "remote image". At the same time, it can be shown that proceeding from (29), (30) at the output of the i-th neuron of the second layer of the neural network, corresponding to the "remote image", the value will always be equal to zero. To do this, we first estimate the value of the neuron of the first layer (Sn_{1,k_r}), which compares the sample with the sample of the "remote image":

$$\begin{aligned} Sn_{1,k\_r} &= \sum_{i=1}^{M}\sum_{j=1}^{K} x_{ij} w_{ij,k} - \sum_{i=1}^{M}\sum_{j=1}^{K} x_{ij} w_r \le \sum_{i=1}^{M}\sum_{j=1}^{K} x_{ij} w_{\max} - \sum_{i=1}^{M}\sum_{j=1}^{K} x_{ij} w_r = \\ &= \sum_{i=1}^{M}\sum_{j=1}^{K} \left(x_{ij} w_{\max} - x_{ij} w_r \right) = \sum_{i=1}^{M}\sum_{j=1}^{K} x_{ij} \left(w_{\max} - w_r \right) = \\ &= \sum_{i=1}^{M}\sum_{j=1}^{K} x_{ij} \left(w_{\max} - Lw_{\max} \right) < 0 \end{aligned} \tag{31}$$

Hence, the activation function $f(Sn_{1,k\_r})$ for a given neuron is equal to 1. In this case, the value of $Sn_{2,r}$ for the neuron of the second layer corresponding to the "remote image" is determined for the case α=1:

$$\begin{aligned} Sn_{2,r} &= \sum_{i=1}^{r-1} \overline{f}(Sn_{1,i\_r}) + \sum_{i=r+1}^{N} f(Sn_{1,r\_i}) = \\ &\sum_{i=1}^{r-1,(i\neq k)} \overline{f}(Sn_{1,i\_r}) + \overline{f}(Sn_{1,k\_r}) + \sum_{i=r+1}^{N} f(Sn_{1,r\_i}) = \\ &= \sum_{i=1}^{r-1,(i\neq k)} \overline{f}(Sn_{1,i\_r}) + \overline{1} + \sum_{i=r+1}^{N} f(Sn_{1,r\_i}) < N-1 \end{aligned} \tag{32}$$

where r is the number of the neuron corresponding to the "remote image" in the zero layer; N is the number of images (samples) of the neural network. Thus, we obtain the output value of the neuron of the second layer for the "remote image" $f(Sn_{2,r}) = 0$. That is, adding a "remote image" does not change the logic of the network.

Paragraf 2.4. the schemes of using a neural network in multitasking are given for the case when the belonging of the input objects to a particular task is not known. In this case, the task of selecting a classifier for tasks is added to the scheme of a neural network for multitasking. For this, it is proposed to add an additional task (classifier) to the neural network (Fig. 23). In this case, to determine this classifier, each task is represented as a separate image (class), for which the sample or class samples are determined, for example, according to the selection algorithm given in [15]. Class weights tables and one “remote image” table for this task are created on the basis of class samples. The procedure for recognizing the input element is carried out according to the scheme in Fig. 23. At the first stage, the required class is determined, which corresponds to the object being recognized (Fig. 23 (1)), for which the class weight tables are fed to the network inputs, and the values of the "remote image" weight table are fed to the unused neurons of the zero layer. In this case, the number of the active output of the neural network, which determines the number of the task to which the object belongs, activates the classifier of the corresponding task through feedback (Fig. 23 (2)). Tables of weights of the selected task are sent to the inputs of the neural network, and the output sample set Y_{out} for this task is sent to the output of the neural network (Fig. 23 (3)), after which the object is recognized within the framework of the selected task.

Paragraf 2.5. an algorithm for sequential application of samples to a neural network based on the metric recognition method is proposed. A scheme and an algorithm for using a neural network in multitasking is given for cases when the number of samples in tasks is greater than the resources of the neural network allow, that is, there are more number of neurons in the zero layer.

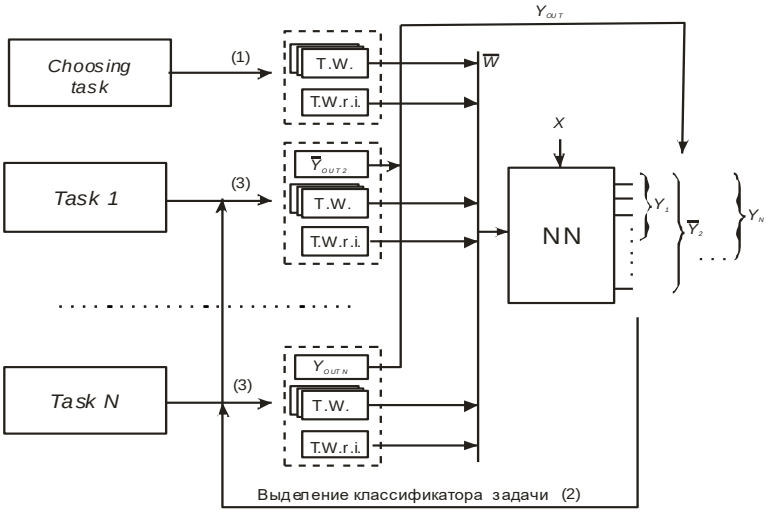


Fig. 23 Network operation diagram with preliminary allocation of the task classifier.

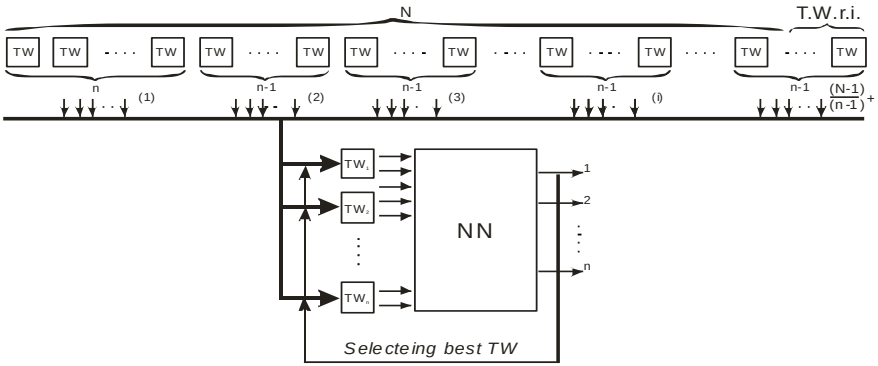


Fig. 24 Scheme of sequential enumeration of samples in the network.

When the condition $n < N_{smp}$ is fulfilled, where n is the number of neurons in the zero layer, N_{smp} is the number of samples used, initially the weight tables of the first n samples are fed to the network. The results at the output of the neural network through feedbacks (Fig. 24) select the *i*-th table of weights ($T.W._i$) of the

closest sampled corresponding to the active output, the values of the weights of which are re-fed to the inputs of the *i*-th neuron of the zero layer.

In this case, tables of weights of the following (n – 1) samples are fed to the remaining (n – 1) neurons of the zero layer. The procedure continues in the same way until all (N_{et}) tables of weighting samples have been used. In this case, if at some stage of the sequential enumeration algorithm (for example, at the last one, Fig. 24), some neurons of the zero layer remain unused, then the table of weights of the "remote image" defined for this task is sent to the non-effective neurons of the zero layer.

Paragraf 2.6. and 2.7. the simplest examples of the implementation of the multitasking application of the neural network based on the metric recognition method are presented. For example, in Fig. 25 shows a diagram of applying the two simplest tasks to one network. Each problem defines two images. For the first problem Y_{out} = {*curve1*, *curve2*}, for the second Y_{out} = {7, 2}.

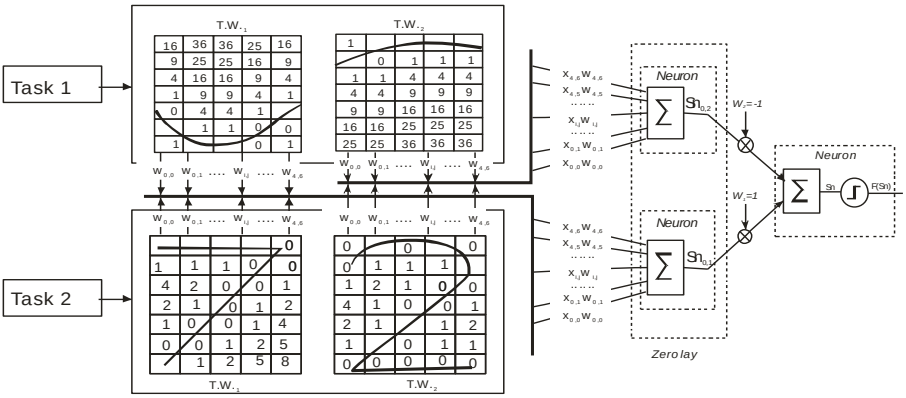


Fig. 25 Scheme of applying a neural network to two problems with two samples.

Paragraf 2.8. provides conclusions regarding chapter II.

Chapter III discusses schemes and algorithms for applying neural networks based on metric recognition methods in decision-making problems with fuzzy inferences [9].

Paragraf 3.1. an algorithm for converting the fuzzy inference problem into the curve recognition problem is proposed. For this, a scheme is proposed for converting fuzzy sets and fuzzy rules of the fuzzy inference problem into a set of curves.

In fig. 26 (*a*, *b*, *c*) shows examples of fuzzy sets x_i defined on ranges Δx_i. Moreover, if fuzzy sets are defined on individual elements of the set, then in this case certain sets of selected elements of sets can be represented as separate sample sets $\overline{X}$.

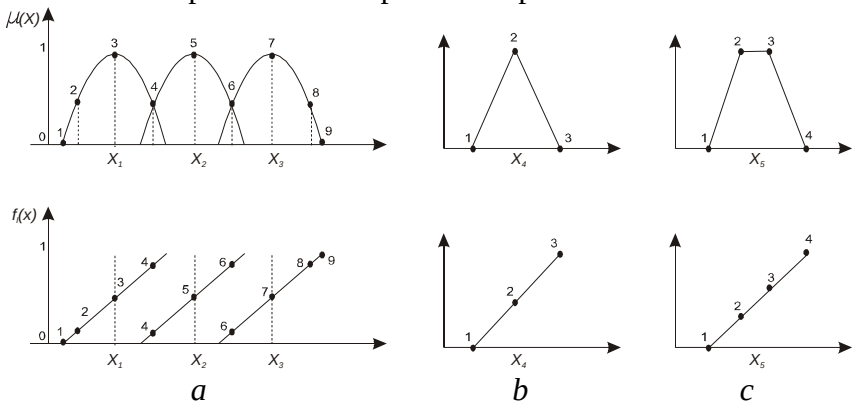


Fig. 26 Converting sets to linear form.

The set shown in Fig. 26*a*, consists of three fuzzy subsets X1, X2, X3, defined on 9 selected elements of the set. Each selected element of the subset has its own value $\mu(x)$. Moreover, if the fuzzy subset X_i is defined on a certain range of values [$x_{i,1}$, $x_{i,2}$], then the linear values $f_l(x_{i,j})$ of the selected elements $x_{i,j}$ are determined from the expression:

$$f_l(x_{i,j}) = \frac{1}{x_{i,2} - x_{i,1}} x_{i,j} + \frac{1}{x_{i,2} - x_{i,1}} x_{i,1} \qquad (33)$$

where $f_l(x_i)$ is a linear function of the *i*-th subset.

If each fuzzy subsets in Fig. 26*a* is represented as points on one point of the *X*-axis of the coordinates of the curve, then we get a curve consisting of 3 points. At the same time, to determine the number of possible curves, we will call two sets (subsets) dependent if each element of one set can be associated with only one element of another set and independent if each element of one set can be associated with any element of another set.

For example, shown in Fig. 26a, subsets are dependent because they are terms of the same set. For such sets, the number of curves will correspond to the number of selected elements of the set. In fig. 27 shows 9 possible curves obtained on the basis of 9 linear values $f_l(x_{i,j})$ of the elements of the set in Fig. 26a.

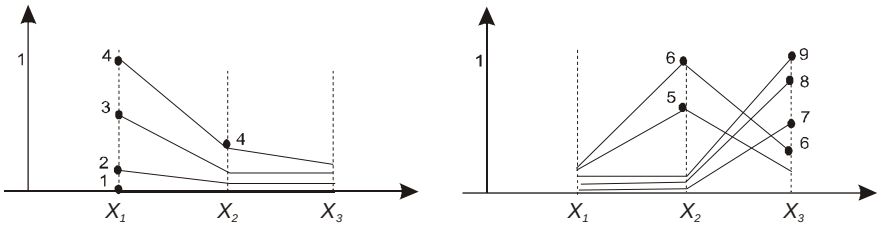


Fig. 27 Sets of curves for subsets X_1, X_2, X_3.

Each sample curve at the output of the recognition system can correspond to one or more output values y_i (Fig. 28).

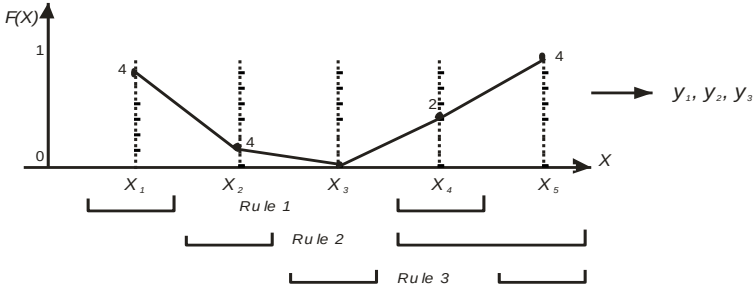


Fig. 28 Example of one sample curve.

Consider an example of a possible set of fuzzy rules in relation to fuzzy sets shown in Fig. 26:

Rule 1: If x is equal to X_1 and x is equal to X_4 then y is equal to Y_1
Rule 2: If x is equal to X_2 and x is equal to X_4 or x is equal to X_5 then y is equal to Y_2
Rule 3: If x is equal to X_3 or x is equal to X_5 then y is equal to Y_3

In fig. 28 it can be seen that the sample curve combines the given 3 fuzzy rules, while the output of the recognition system of the sample curve corresponds to three sample output values y_1, y_2, y_3 corresponding to the sets Y_1, Y_2, Y_3 (Fig. 28). Similarly, the resulting curves can include possible other rules based on the sets from which the curves were created.

Paragraf 3.2. a diagram of the application of a neural network based on metric recognition methods to the problem of recognizing the obtained set of curves and making the final solution of a fuzzy inference based on the proposed neural networks is given (fig. 29, 31).

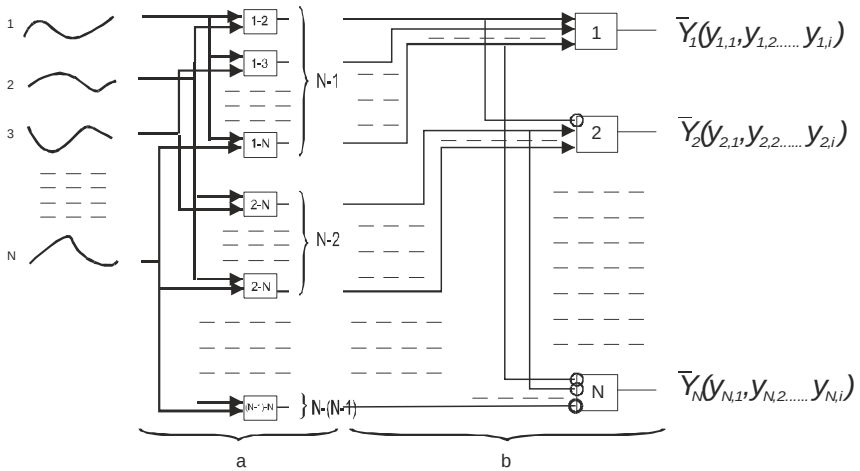


Fig. 29 Neural network for *N* images.

Each *i*-th output of the neural network in fig. 29 will correspond to the output vector $\bar{Y}_i$, the values of which will be

activated if the value of the corresponding output of the neural network is equal to 1. As the values of the output vector of the neural network in fig. 29 can be given both the values of $f_l(y_{i,j})$ linear functions for all output sets, and directly the values of the $\overline{Y}_i$ output sets themselves, determined according to the applied fuzzy inference algorithm, or from expression (33) if the linear values of the function are known $f_l(y_{i,j})$:

$$y_{i,j} = (y_{i,2} - y_{i,1})\left(f_l(y_{i,j}) - \frac{1}{y_{i,2} - y_{i,1}} y_{i,1} \right) \tag{34}$$

To obtain more accurate values Y_i, it is necessary to determine the intermediate results of the output values, the value of which can be determined based on the values of the S_k proximity coefficients.

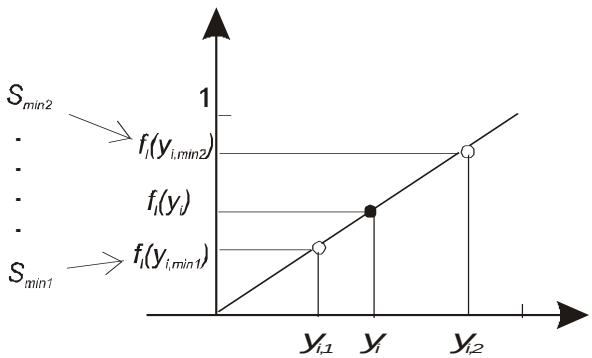


Fig. 30 Determination of intermediate values.

For this, in relation to the tested curve, the values of S_k are determined for the nearest two sample curves, $S_{\min 1}$ and $S_{\min 2}$ (Fig. 30). In this case, the output value of the $f_l(y_i)$ linear function will be determined from the expression:

$$\frac{f_l(y_{i,\min 1}) - f_l(y_i)}{f_l(y_i) - f_l(y_{i,\min 2})} = \frac{\sqrt{S_{\min 1}/N}}{\sqrt{S_{\min 2}/N}}, \tag{35}$$

where is $f_l(y_i)$ - the final intermediate value of the linear function for the *i*-th output variable, $f_l(y_{i,\min 1})$, $f_l(y_{i,\min 2})$ - are the values of the linear functions for the *i*-th output value corresponding to the nearest two sample curves at the outputs of the neural network (Fig. 29), N - is the number of curve counts. Since the f_l function is linear with respect to its variable, in expression (35), the value of the f_l function can be replaced by the value of the variables:

$$\frac{y_{i,\min 1} - y_i}{y_i - y_{i,\min 2}} = \frac{\sqrt{S_{\min 1}/N}}{\sqrt{S_{\min 2}/N}}, \text{ then}$$

$$y_i = \frac{y_{i,\min 1}\sqrt{S_{\min 2}} + y_{i,\min 2}\sqrt{S_{\min 1}}}{\sqrt{S_{\min 1}} + \sqrt{S_{\min 2}}} \qquad (36)$$

Expression (36) defines the output values for all output values of fuzzy sets, that is, the resulting output values are defasified. To implement expression (36), it is necessary to determine $S_{\min 1}$, $S_{\min 2}$ the values of which can be obtained directly from the neural network. For this, a zero layer (Fig. 31a) is added to the neural network (Fig. 29), consisting of linear neurons, the activation functions of which are equal to the values of the neurons themselves (15).

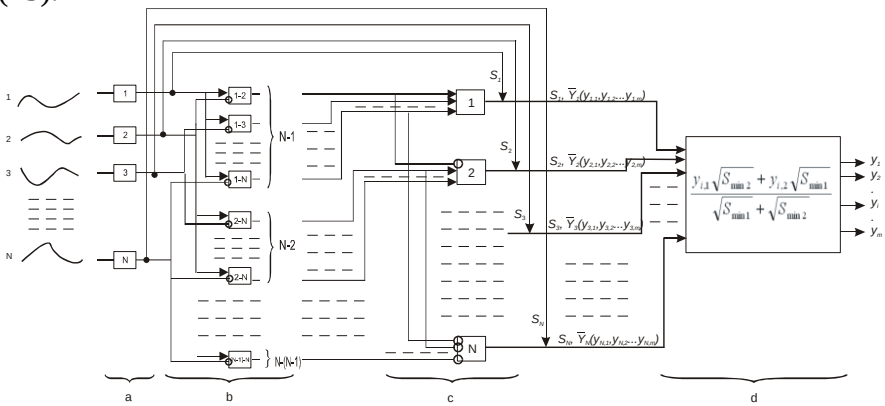


Fig. 31 Neural network and defasification block for *N* images.

Paragraf 3.3. the possibilities of simplifying and minimizing the circuit in fig. 29, 30 are considered in order to reduce the number of neurons and the dimension of the weight tables. For this purpose, it is proposed to divide a set of curves into ranges of curves determined by fuzzy rules and thus transform a single neural network based on the metric recognition method into a multi-block structure of a neural network (fig. 32). With the aim of further minimizing the neural network, the application of one block neural network in a multitasking application for all fuzzy rules is also considered.

Paragraf 3.4. an algorithm is proposed for selecting expert opinions in relation to a neural network based on the metric recognition method.

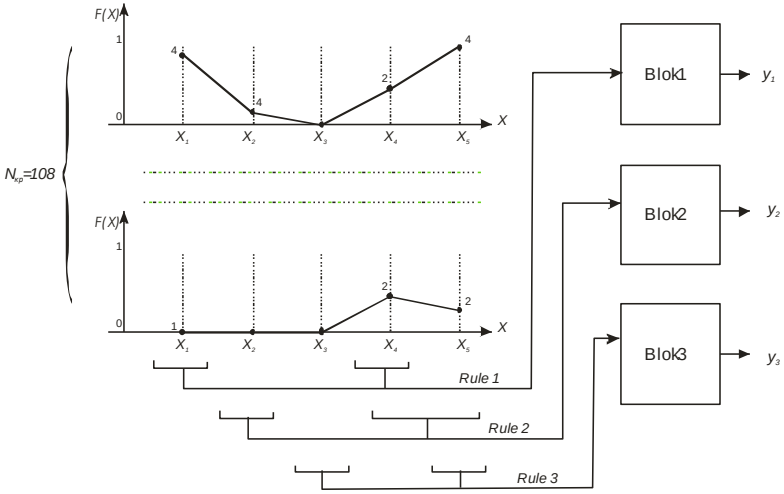


Fig. 32 Dividing the network into separate blocks.

Paragraf 3.5. summarizes the conclusions of chapter III.

Chapter IV transforms a neural network based on the metric recognition method into a fully connected multilayer perceptron [18, 19]. For this purpose, the regularities determining the distributions of the weight and threshold values of the neural network are investigated on the basis of metric recognition methods. A

generalized algorithm for calculating the weight and threshold values of a fully connected multilayer perceptron based on the metric recognition method is presented. It is shown that a neural network based on metric recognition methods is a special case of a multilayer perceptron, and, accordingly, can be trained by classical learning algorithms. At the same time, in contrast to conventional multilayer perceptron architectures, the resulting multilayer perceptron retains all the advantages of a neural network based on the metric recognition method.

Paragraf 4.1. the general statement of the problem is given, as well as the rationale for the feasibility of solving this problem.

Paragraf 4.2. generalized patterns of distribution of values of weight and threshold values for the second layer of a neural network are investigated based on metric recognition methods. It was shown that for the distribution of the input weight $w_{k,j}^{(2)}$ and threshold $H_k^{(2)}$ values of the *k*-th neuron of the second layer for the scheme of an incompletely connected network in fig. 2, the following condition must be satisfied.

$$\sum_{\substack{j=1 \\ (j\neq k)}}^{N-1} w_{k,j}^{(2)} - w_{k,j\ min}^{(2)} < H_k^{(2)} \leq \sum_{\substack{j=1 \\ (j\neq k)}}^{N-1} w_{k,j}^{(2)} \qquad (37)$$

where $w_{k,j\ min}^{(2)}$ - min is the minimum input value $w_{k,j}^{(2)}$ of the *k*-th neuron of the second layer.

It was also shown that to fulfill the condition of the activation function (5) of the *k*-th neuron of the second layer, it is also necessary to fulfill the conditions $w_{k,j}^{(2)} > 0$ for all *j* inputs of the neuron of the second layer.

$$\begin{cases} f\left(Sn_k^{(2)}\right) = 1, \text{ if } Sn_k^{(2)} \geq H_k^{(2)} \\ f\left(Sn_k^{(2)}\right) = 0, \text{ if } Sn_k^{(2)} < H_k^{(2)} \end{cases} \qquad (38)$$

where $Sn_k^{(2)}$ - is the state function of the *k*-th neuron of the second layer.

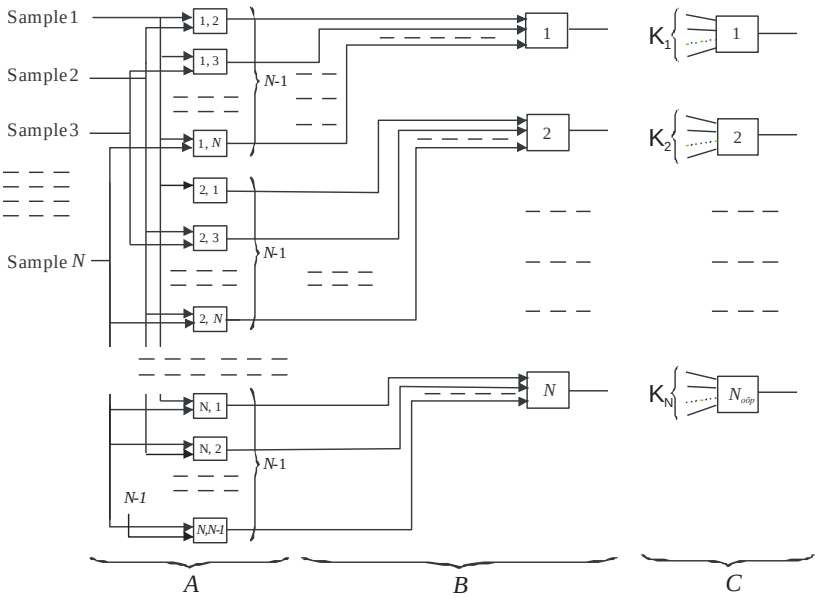


Fig. 33 Three-layer neural network based on metric recognition methods with non-minimized architecture.

It was shown that for the case of transforming the neural network in fig. 33 into a fully connected neural network, the state values of the *k*- th neuron of the second layer are transformed into the expression:

$$Sn_k^{(2)} = \sum_{i=1}^{N} \sum_{\substack{j=1 \\ (j \neq i)}}^{N-1} \left(w_{i,j}^{(2)} f\left(Sn_{i,j}^{(1)} \right) \right), \qquad (39)$$

where *i*, *j* determine the serial numbers of the pairwise shared samples (fig. 33*a*). At the same time, it was also shown that for the kth neuron of the second layer for all added connections *(i ≠ k),* in order to maintain the stability of the activation function (5), it is also necessary to fulfill the conditions (40, 41):

$$\sum_{\substack{i=1, \\ i \neq k}}^{N-1} \sum_{\substack{j=1, \\ j \neq i, \\ w_{i,j}^{(2)} \geq 0}}^{N-1} w_{i,j}^{(2)} < H_k^{(2)} - \left(\sum_{\substack{j=1, \\ j \neq k}}^{N-1} \left(w_{k,j}^{(2)} f\left(Sn_{k,j}^{(1)} \right) \right) - w_{k,j\ min}^{(2)} \right), \qquad (40)$$

$$\left|\sum_{\substack{i=1,\\ i\neq k}}^{N-1}\sum_{\substack{j=1,\\ j\neq i,\\ w_{i,j}^{(2)}<0}}^{N-1} w_{i,j}^{(2)}\right| \leq \sum_{\substack{j=1,\\ J\neq k}}^{N-1}\left(w_{k,j}^{(2)} f\left(Sn_{k,j}^{(1)}\right)\right) - H_k^{(2)}, \qquad (41)$$

where the left sides in expressions (40), (41) are the positive and negative components of the sum of all $w_{i,j}^{(2)}$ for the *k*-th neuron of the second layer, for which $i \neq k$. Failure to comply with conditions (40, 41) will mean the possibility of the appearance of erroneous values at the output of the *k*-th neuron of the second layer.

Paragraf 4.3. the regularities of distribution of weight and threshold values of the third layer of the neural network are investigated. It was shown that for the condition of the activation function of the third layer to be met (10):

$$\begin{cases} f\left(Sn_k^{(3)}\right) = 1, \text{ if } Sn_k^{(3)} \geq H_k^{(3)} \\ f\left(Sn_k^{(3)}\right) = 0, \text{ if } Sn_k^{(3)} < H_k^{(3)} \end{cases} \qquad (42)$$

it is necessary that the threshold value of the *k*- th neuron of the third layer $H_k^{(3)}$ for an incompletely connected neural network is determined in the range (43):

$$0 < H_k^{(3)} \leq w_{k,j}^{(3)(\min)} \qquad (43)$$

where $w_{k,j}^{(3)(\min)}$ is the minimum value of the weight $w_{k,j}^{(3)}$ for the *k*-th neuron of the third layer.

It was also shown that for a fully connected neural network for the stability of the third layer, the values of the weights of the added connections $w_{k,j}^{(3)}$ for $j \neq j_k$ must be determined by the conditions:

$$\sum_{\substack{j=1,\\ j\neq j_k,\\ w_{k,j}^{(2)}\geq 0}}^{N} w_{k,j}^{(3)} < H_k^{(3)} \qquad (44)$$

$$\left| \sum_{\substack{j=1, \\ j \neq j_k, \\ w_{k,j}^{(2)} < 0}}^{N} w_{k,j}^{(3)} \right| \leq w_{k,j_k}^{(3)(min)} - H_k^{(3)}, \qquad (45)$$

where j_k are the j-th inputs of the k-th neuron of the third layer, connecting to the neurons of the second layer, the outputs of which correspond to the samples of the k-th image, $w_{k,j_k}^{(3)(min)}$ is the minimum weight value for j_k connections. The left sides of expressions (44, 45) are the positive and negative components of the sum of weights for all added connections of the k-th neuron of the third layer ($j \neq j_k$).

Paragraf 4.4. based on the above studies, a general algorithm for determining the distribution of threshold and weight values for the perceptron, created on the basis of the metric recognition method, was given. In particular, the proposed algorithm for determining the weight and threshold values of the second layer of a fully connected neural network is as follows:

1. On the basis of the existing samples set, a network is built according to the scheme in fig. 33, into which links are further added, as a result of which the network is transformed into a fully connected multilayer perceptron. In this case, the number of neurons for the first layer is determined by expression (7), the number of neurons in the second layer is determined by the number of samples, the number of neurons in the third layer is determined by the number of recognized patterns.

2. For the k-th neuron, the values of the weights for all added connections $w_{i,j}^{(2)}$ (for which $i \neq k$) are chosen randomly.

3. From the obtained values $w_{i,j}^{(2)}$, the values of the sums S_1 and S_2 of positive and negative components, taken modulo, are determined:

$$S_1 = \sum_{\substack{i=1 \\ i\neq k}}^{N-1} \sum_{\substack{j=1, \\ j\neq i, \\ w_{i,j}^{(2)}\geq 0}}^{N-1} w_{i,j}^{(2)}, S_2 = \left| \sum_{\substack{i=1 \\ i\neq k}}^{N-1} \sum_{\substack{j=1, \\ j\neq i, \\ w_{i,j}^{(2)}<0}}^{N-1} w_{i,j}^{(2)} \right| \quad (46)$$

4. Determine the minimum value $w_{k,j\ min}^{(2)}$, according to the expression:

$$w_{k,j\ min}^{(2)} = S_1 + S_2 + \beta \quad (47)$$

where β - is any positive number, $\beta \in (0, \infty)$. In this case, expression (47) is determined on the basis of adding the right and left sides of expressions (44) and (45) and reducing the inequality to equality:

$$\sum_{\substack{i=1, \\ i\neq k}}^{N-1} \sum_{\substack{j=1, \\ j\neq i, \\ w_{i,j}^{(2)}\geq 0}}^{N-1} w_{i,j}^{(2)} + \left| \sum_{\substack{i=1,, \\ i\neq k}}^{N-1} \sum_{\substack{j=1, \\ j\neq i, \\ w_{i,j}^{(2)}<0}}^{N-1} w_{i,j}^{(2)} \right| << H_k^{(2)} - \left(\sum_{\substack{j=1, \\ j\neq k}}^{N-1} \left(w_{k,j}^{(2)} f\left(Sn_{k,j}^{(1)} \right) \right) - w_{k,j\ min}^{(2)} \right) + + \sum_{\substack{j=1, \\ J\neq k}}^{N-1} \left(w_{k,j}^{(2)} f\left(Sn_{k,j}^{(1)} \right) \right) - H_k^{(2)} \quad (48)$$

$$\sum_{\substack{i=1 \\ i\neq k}}^{N-1} \sum_{\substack{j=1, \\ j\neq i, \\ w_{i,j}^{(2)}\geq 0}}^{N-1} w_{i,j}^{(2)} + \left| \sum_{\substack{i=1, \\ i\neq k}}^{N-1} \sum_{\substack{j=1, \\ j\neq i, \\ w_{i,j}^{(2)}<0}}^{N-1} w_{i,j}^{(2)} \right| < w_{k,j\ min}^{(2)} \quad (49)$$

$$\sum_{\substack{i=1, \\ i\neq k}}^{N-1} \sum_{\substack{j=1, \\ j\neq i, \\ w_{i,j}^{(2)}\geq 0}}^{N-1} w_{i,j}^{(2)} + \left| \sum_{\substack{i=1, \\ i\neq k}}^{N-1} \sum_{\substack{j=1, \\ j\neq i, \\ w_{i,j}^{(2)}<0}}^{N-1} w_{i,j}^{(2)} \right| + \beta = w_{k,j\ min}^{(2)} \quad (50)$$

5. We randomly choose the remaining values $w_{k,j}^{(2)}$, while observing the condition $w_{k,j}^{(2)} \geq w_{k,j\ min}^{(2)}$, and also determine the sum

S_k of all $w_{k,j}^{(2)}$.

$$S_k = \sum_{\substack{j=1,\\ j\neq k}}^{N-1} \left(w_{k,j}^{(2)} f\left(Sn_{k,j}^{(1)} \right) \right) \tag{51}$$

6. Determine the values of the threshold $H_k^{(2)}$ for this neuron:

$$H_k^{(2)} = \left(S_k - w_{k,j\ min}^{(2)} \right) + S_1 + c \tag{52}$$

where $c \in (0;\ \beta]$

7. Go to the next k+1 neuron and continue the algorithm from step 2.

For the third layer, the algorithm for determining the values of $w_{k,j}^{(3)}$ and $H_k^{(3)}$ will differ in that in step 4 the minimum value $w_{k,j_k}^{(3)(min)}$, as

$$w_{k,j_k}^{(3)(min)} = S_1 + S_2 + \beta\ , \tag{53}$$

where

$$S_1 = \sum_{\substack{j=1,\\ j\neq j_k,\\ w_{k,j}^{(2)}\geq 0}}^{N} w_{k,j}^{(3)}\ , S_2 = \left| \sum_{\substack{j=1,\\ j\neq j_k,\\ w_{k,j}^{(2)}<0}}^{N} w_{k,j}^{(3)} \right|, \tag{54}$$

and the threshold value $H_k^{(3)}$ for the k-th neuron of the third layer is determined from expression (55), as

$$H_k^{(3)} = S_1 + c \tag{55}$$

where $c \in (0;\ \beta]$.

Paragraf 4.5. the mathematical substantiation of the distributions of threshold and weight values is given on the example of using the learning algorithm according to the Widrow Hoff method. Based on this learning algorithm, it is shown that when training a fully connected multilayer perceptron, the distribution of the values of the weight and threshold values tends to satisfy conditions (40), (41), (43), (44), (45), (49).

Paragraf 4.6. the simplest example is presented that demonstrates the creation of a multilayer perceptron with a complete

system of connections and with a threshold activation function with calculated parameters. As the simplest example, the problem of recognition of two handwritten graphic images {4, 3}, consisting of a set of three samples, shown in fig. 34, is given.

Fig. 34 An example of a set of samples for two images.

In the example, a general algorithm was used to determine the distribution of threshold and weight values for the second (Tab. 5) and third layer (Tab. 6) of a multilayer perceptron, created on the basis of the metric recognition method.

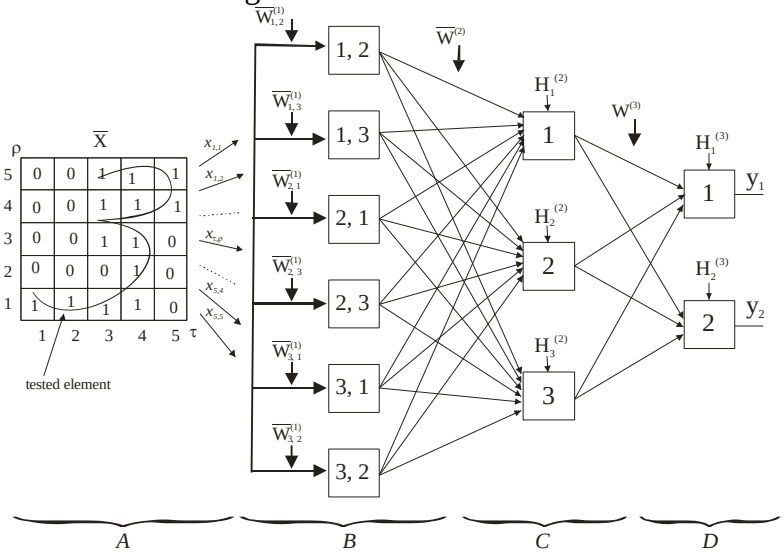


Fig. 35 Recognition scheme using a three-layer neural network, (a) tested element $\overline{X}$ (b) first layer, (c) second layer, (d) third layer.

Tab 5. The value of the weights $w_{i,j}^{(2)}$ and the algorithm parameters for the second layer.

$w_{i,j}^{(2)}$ for the k-th neuron							Algorithm parameters for the 2nd layer						
k \ i, j	1, 2	1, 3	2, 1	2, 3	3, 1	3, 2	S_1	S_2	β	$w_{k,j\ min}^{(2)}$	S_k	c	$H_k^{(2)}$
1	19	15	-2	-2	3	4	7	4	4	15	34	3	29
2	-4	2	11	14	0	3	5	4	2	11	25	1	20
3	-5	3	2	5	22	17	10	5	2	17	39	1	33

Tab. 6 Value of weights $w_{k,j}^{(3)}$ and algorithm parameters for the third layer.

$w_{k,j}^{(3)}$ for the k-th neuron				Algorithm parameters for the 3nd layer						
k \ j	1	2	3	S_1	S_2	β	$w_{k,j_k}^{(3)(min)}$	S_k	c	$H_k^{(3)}$
1	*7*	2	5	2	0	3	5	12	2	4
2	1	9	-4	1	4	4	9	9	1	2

Tab. 7 Values of neurons and activation functions for element $\overline{X}$ in fig. 35.

first layer				second layer				third layer				$Y_{вых}$
№ (i, j)	$Sn_{i,j}^{(1)}$	$f(Sn_{i,j}^{(1)})$		№ k	$Sn_k^{(2)}$	$f(Sn_k^{(2)})$		№ k	$Sn_k^{(3)}$	$y_k = f(Sn_k^{(3)})$		
1, 2	-18	≥ 0	0	1	-1	≥29	0	1	2	≥4	0	-
1, 3	-14	≥ 0	0	2	25	≥20	1	2	9	≥2	1	{3}
2, 1	18	≥ 0	1	3	29	≥33	0					
2, 3	3	≥ 0	1									
3, 1	14	≥ 0	1									
3, 2	-3	≥ 0	0									

An example (Tab. 7) of recognition of the tested character "3" (Fig. 35) was also given, which demonstrated that with these

calculated parameters, the neural network works correctly and identifies the tested character of the handwritten digit "3".

In table 7, the values of the states of neurons $Sn_{i,j}^{(1)}$ were determined by expression (3), where the values of the weights $w_{\tau,\rho}^{(1)}$ for the first layer were determined by (13), (14) the proximity measure (Fig. 36).

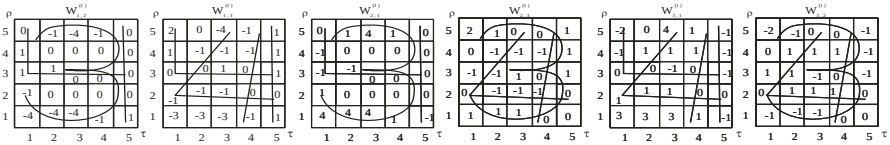


Fig. 36 Weight tables for the first layer of the neural network.

Paragraf 4.6. the mathematical substantiation of the distributions of threshold and weight values is given on the example of using the learning algorithm according to the Widrow Hoff method. Based on this learning algorithm, it is shown that when training a fully connected multilayer perceptron, the distribution of values of weight and threshold values tends to fulfill the conditions (16), (19), (20), (22), (23), (24).

Paragraf 4.7. an example of a comparative analysis based on MNIST of learning outcomes for a classical multilayer perceptron and perceptron implemented on the basis of the nearest neighbor method is given. The MNİST database includes 60,000 digits of the training input and output number of digit patterns and 10,000 tested input and output number of digit patterns of characters corresponding to the images of digits from 0 to 9. The example was implemented in the Python software environment using the *Keras* module built into this environment. To create a neural network, a random from the MNIST base 30 samples (3 samples from each digit image). The structure of the neural network was determined according to the scheme shown in fig. 33, with further transformation into a fully connected multilayer perceptron. As a result, the number of neurons in the first layer was 870 neurons, in the second layer - 30 neurons, in the third layer - 10 neurons. A stochastic gradient descent

algorithm was used as the learning algorithm. To estimate the direction of the descent gradient, 200 symbols were used. The comparative analysis was carried out on the basis of two experiments. The resulting neural network was first trained in the classical way, by the chosen learning algorithm - without calculating the weight and threshold values. As a result of this training given below, the neural network at the 100th epoch achieved a result of 96.9% on the training set and 95% on the tested set:

Train on 54000 samples, validate on 6000 samples Epoch 1/10054000/54000
[==========] - 9s - loss: 1,2561- acc: 0,7101 - val_loss: 1,3643 - val_acc: 0,6712 Epoch 1/10054000/54000
[==========] - 9s - loss: 0,7112- acc: 0,8392 - val_loss: 0,7344 - val_acc: 0,8235 Epoch 2/10054000/54000
[==========] - 11s - loss: 0,5547- acc: 0,8792 - val_loss: 0,5763 - val_acc: 0,8689 Epoch 3/10054000/54000
……
……
[==========] - 12s - loss: 0,1121- acc: 0,9803 - val_loss: 0,1181 - val_acc: 0,9686 Epoch 99/10054000/54000
[==========] - 12s - loss: 0,1175- acc: 0,9822 - val_loss: 0,1175- val_acc: 0,9691 Epoch 100/10054000/54000
[==========] - 11s - loss: 0,1191- acc: 0,9728 - val_loss: 0,1209- val_acc: 0,9502 10000/10000

For the second experiment, the same neural network structure was used. In this case, the values of the weights of the first layer were first calculated by expression (35). Weight and threshold values

were scaled to the range from [-1, 1]. The weight and threshold values of the second and third layers were set to 0 or 1, according to the scheme shown in fig. 37.

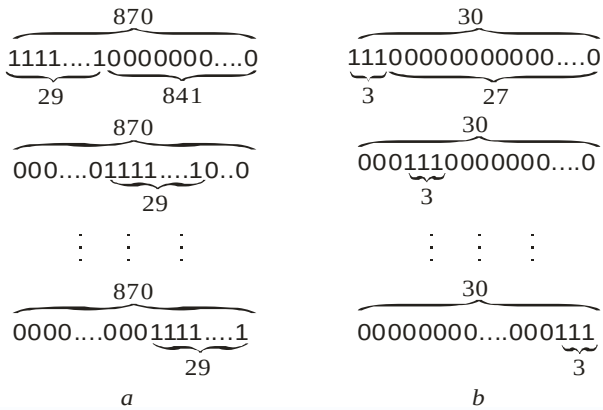


Fig. 37 (a) Value of weights for the second layer, (b) Value of weights for the third layer.

The circuit was then trained by a stochastic gradient descent algorithm. The training results are shown in the listing:

[==========] - 9s - loss: 0.2293 - acc: 0.9679 - val_loss: 0.1094 - val_acc: 0.9604Epoch1/2054000/54000
[==========] - 9s - loss: 0.0814 - acc: 0.9753 - val_loss: 0.0918 - val_acc: 0.9711 Epoch 2/2054000/54000
[==========] - 11s - loss: 0.0534 - acc: 0.9867 - val_loss: 0.0773 - val_acc: 0.9775 Epoch 3/2054000/54000
[==========] - 12s - loss: 0.0356 - acc: 0.9879 - val_loss: 0.0622 - val_acc: 0.9807 Epoch 4/2054000/54000
[==========] - 12s - loss: 0.0243 - acc: 0.9912 - val_loss: 0.0739 - val_acc: 0.9779 Epoch 5/2054000/54000

[==========] - 12s - loss: 0.0221 - acc: 0.9941 - val_loss: 0.0753 - val_acc: 0.9815 Epoch 6/2054000/54000
[==========] - 12s - loss: 0.0212 - acc: 0.9926 - val_loss: 0.0845 - val_acc: 0.9774 Epoch 7/2054000/54000
[==========] - 13s - loss: 0.0174 - acc: 0.9915 - val_loss: 0.0884 - val_acc: 0.9772 Epoch 8/2054000/54000
[==========] - 14s - loss: 0.0128 - acc: 0.9948 - val_loss: 0.0833 - val_acc: 0.9756 Epoch 9/2054000/54000
[==========] - 13s - loss: 0.0118 - acc: 0.9977 - val_loss: 0.0834 - val_acc: 0.9820 Epoch 10/2054000/54000
[==========] - 13s - loss: 0.0121 - acc: 0.9965 - val_loss: 0.0863 - val_acc: 0.9795 Epoch 11/2054000/54000
[==========] - 13s - loss: 0.0169 - acc: 0.9954 - val_loss: 0.0781 - val_acc: 0.9836 Epoch 12/2054000/54000
[==========] - 12s - loss: 0.0091 - acc: 0.9966 - val_loss: 0.1015 - val_acc: 0.9811 Epoch 13/2054000/54000
[==========] - 12s - loss: 0.0071 - acc: 0.9979 - val_loss: 0.0920 - val_acc: 0.9814 Epoch 14/2054000/54000
[==========] - 12s - loss: 0.0111 - acc: 0.9969 - val_loss: 0.1177 - val_acc: 0.9773 Epoch 15/2054000/54000
[==========] - 12s - loss: 0.0115 - acc: 0.9967 - val_loss: 0.0831 - val_acc: 0.9808 Epoch 16/2054000/54000

[==========] - 12s - loss: 0.0073 - acc: 0.9976 - val_loss: 0.0807 - val_acc: 0.9835 Epoch 17/2054000/54000
[==========] - 13s - loss: 0.0089 - acc: 0.9972 - val_loss: 0.0924 - val_acc: 0.9825 Epoch 18/2054000/54000
[==========] - 12s - loss: 0.0073 - acc: 0.9975 - val_loss: 0.0804 - val_acc: 0.9836 Epoch 19/2054000/54000
[==========] - 12s - loss: 0.0079 - acc: 0.9969 - val_loss: 0.0815 - val_acc: 0.9831 Epoch 19/2054000/54000
[==========] - 11s - loss: 0.0055 - acc: 0.9975 - val_loss: 0.1212 - val_acc: 0.980210000/10000

From the above listing, you can see that at the 20th epoch the result is 98%. This result is 2% higher than the result of the previous experiment, and besides, the result was obtained 5 times faster than the previous experiment (100/20 = 5). In addition, according to the results of the above training listing, you can see that after the first training epoch, the result on the test data reaches 96%.

Paragraf 4.8. summarizes the conclusions regarding the IV-th chapter.

Chapter V discusses ways and approaches to apply the proposed architectures to various problems with different objects of recognition, such as symbols, photography, texts, etc. Ways of representation and processing of objects are considered before feeding them to the inputs of the proposed neural networks [1-7, 11, 16, 17, 19, 20].

Paragraf 5.1. the algorithm for selecting objects in the image is considered. This approach can allow the use of neural networks for object recognition in photographs and scenes [7, 11].

In paragrafs 5.2., 5.3., 5.4., 5.5., 5.6. algorithms for the selection, processing and presentation of images of symbols (Fig. 38)

and texts are considered before feeding the selected objects into the neural network [1-6].

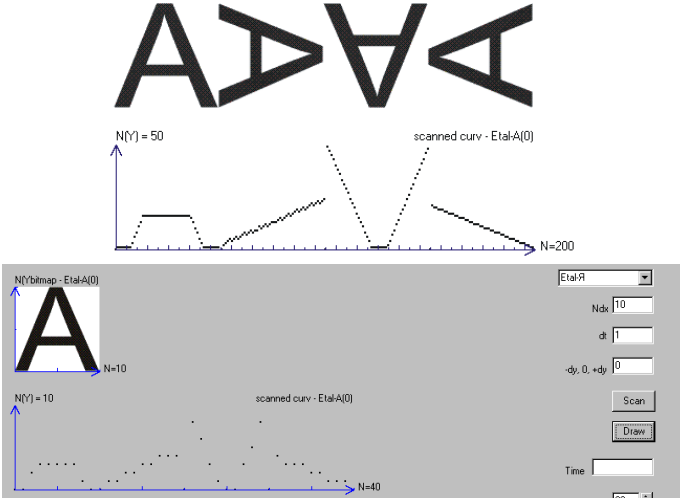


Fig. 38 Examples of conversion curves for the printed character "A" with different sampling steps.

Paragraf 5.7. methods and possibilities of representing symbols with slopes, shifts, rotations, which can be used for the proposed neural networks, are considered [19, 20].

Paragraf 5.8. approaches to solving the problem of face recognition by neural networks based on metric recognition methods are considered. A method of presenting facial images for presentation to the inputs of a neural network is also given [19].

Paragraf 5.9. the possibility of creating a three-dimensional table of weights is considered, which can be used, for example, to represent grayscale or color images, which is necessary, for example, in the task of photographic recognition [19].

Paragraf 5.10., 5.11, 5.12 the methods and approaches of using a neural network based on the metric recognition method for the problem of text content analysis are considered [10, 12, 13, 16, 17, 24]. In particular, an example is considered of analyzing the texts of articles to determine the topics of articles in order to select council

members for the defense of dissertations at an electronic scientific seminar [10, 13, 16, 24]. A method of submission of articles based on keywords suitable for feeding to the inputs of a neural network is considered. The neural network performs the selection of the authors of the necessary topics for the scientific council. As an example of text analysis using the proposed architectures of neural networks, the paragraf also considers an example of determining the addressee of a state organization [12]. The address is determined by the content of the text of the letter of the application, which is necessary for the task of automating the procedure for applying for citizens to a state institution, which allows automating the operator's work of selecting and sorting emails. For this task, the neural network selects the name of the required state organization based on the content of the letter.

In Chapter VI, based on the architectures of neural networks that implement metric recognition methods, the architecture of a neural network with pairwise sequential separation of images is proposed [22, 23]. In this case, the neural network is divided into blocks, each of which, in fact, performs the work of separation - recognition of two patterns. This approach simplifies the procedure for expanding the neural network and adding new patterns to the recognition problem without changing the previous weight and threshold values of the neural network.

For this purpose, in paragraf 6.1. the formulation of this problem is presented.

Paragraf 6.2. the architecture of a two-layer neural network with pairwise sequential separation of images is given, (Fig. 39) [22]. In this case, the first layer of the neural network in the diagram in fig. 39 is a separate block of neural networks NN_{ij}, each of which consists of two or three layers of neural networks (Fig. 40) and each block performs the separation of two images. The union of all the results of the outputs of the blocks NN_{ij} occurs in the second layer (fig. 39*b*). Figure 39*a* also shows a scheme for feeding the input data of a training set to a neural network using the example of a digit recognition problem.

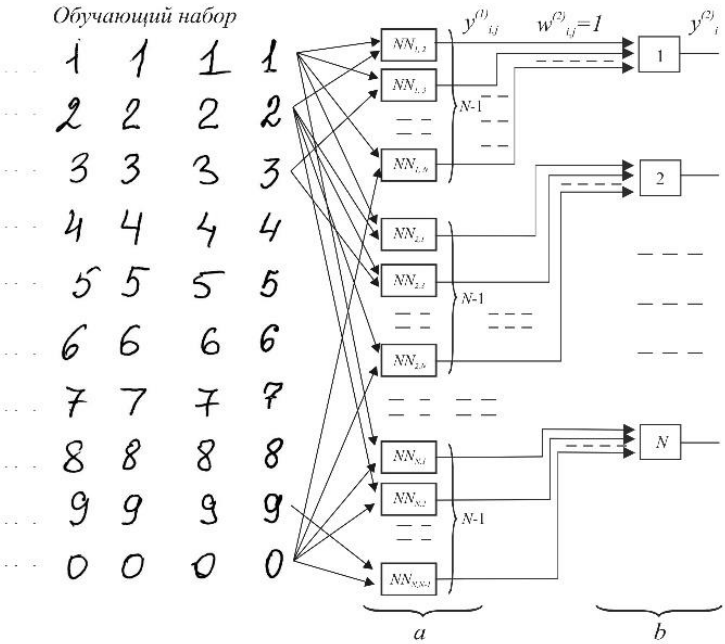


Fig. 39 Neural network training scheme.

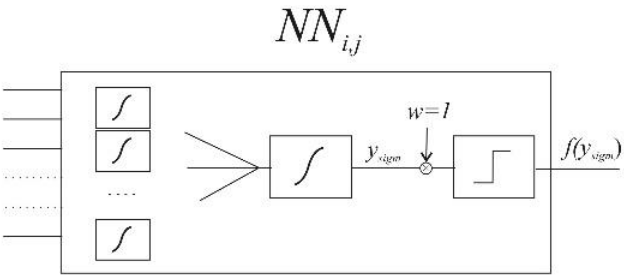


Fig. 40 Converting the output $NN_{i,j}$ into binary form.

Each block of a neural network can include continuous and threshold neurons (Fig. 40). In this case, the continuous analog

output of the neuron block is converted into a discrete value {0, 1} using the output neuron with a threshold activation function, (Fig. 40), according to the expression:

$$f(y_{sigm}) = 1, if\ y_{sigm} > 0{,}5$$
$$f(y_{sigm}) = 0, if\ y_{sigm} \leq 0{,}5 \qquad (56)$$

Paragraf 6.3. the scheme of compression of the obtained neural network in order to reduce the obtained blocks of the neural network is presented. By removing the blocks *NNij* performing the division of the same-name images *{i, j}*, *{j, i}*, the number of blocks is halved (Fig. 41) [22].

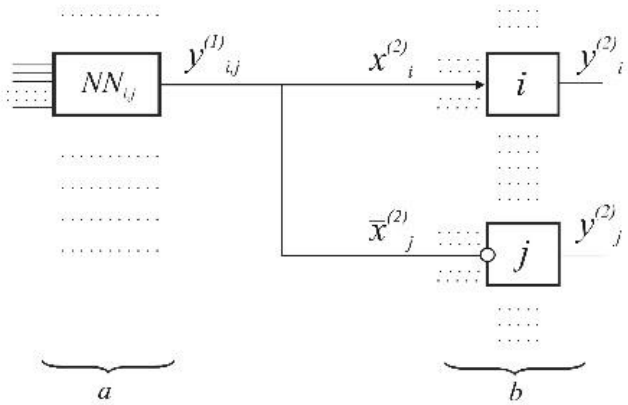


Fig. 41 Neural network compression by excluding the NN_{ji} block, (a) - the first layer, (b) - the second layer.

Paragraf 6.4. an example of creating a neural network with pairwise sequential separation of images is given on the example of the simplest problem of recognizing three patterns {A, B, C}. It also provides formulas for calculating the network operation error for the proposed networks [23].

Paragraf 6.5. conclusions are drawn.

Chapter VII compares the training results of the neural network with the calculated weights and randomly generating the weights

based on MNIST. For this purpose, two experiments are carried out, the results are compared [27, 28].

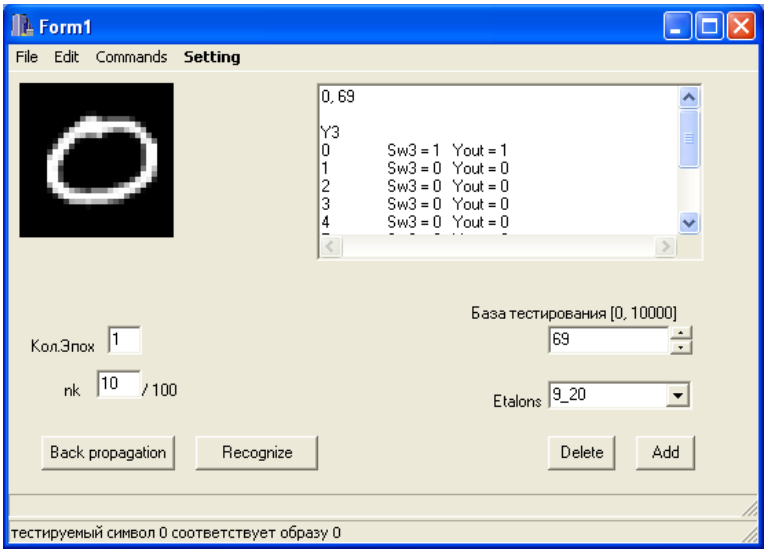


Fig. 42 Software module for recognition and training of images from the MNIST base, implemented in the Builder C ++ environment.

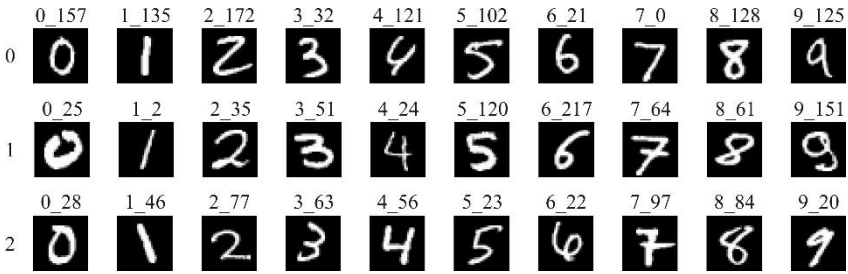
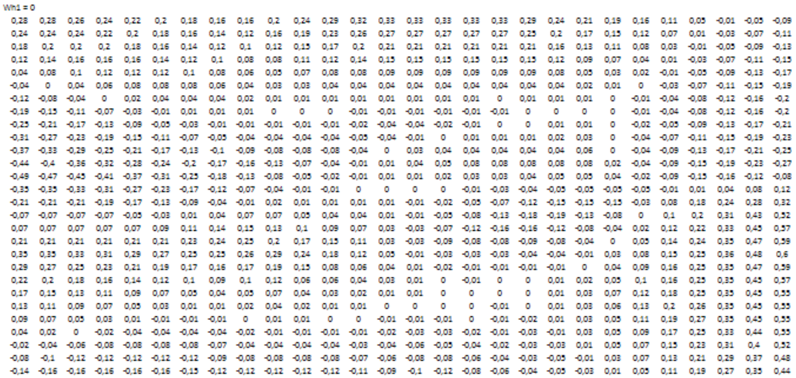


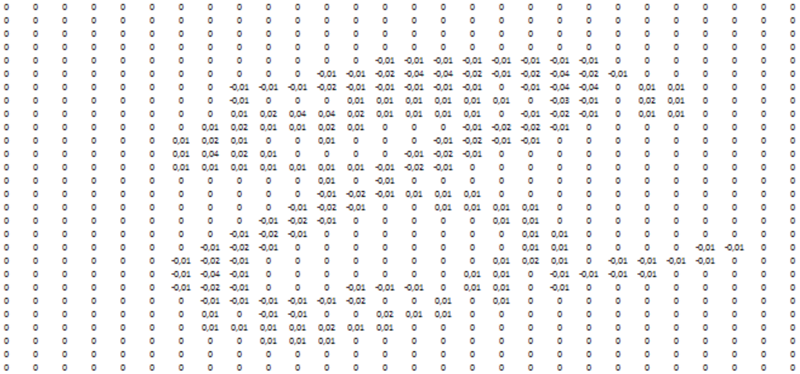
Fig. 43 Selected samples from the control base MNIST, the image name includes the name of the image and the serial number of the image in the MNIST database.



Wh1 = 0

0,28	0,28	0,26	0,24	0,22	0,2	0,18	0,16	0,16	0,2	0,24	0,29	0,32	0,33	0,33	0,33	0,33	0,33	0,29	0,24	0,21	0,19	0,16	0,11	0,05	-0,01	-0,05	-0,09
0,24	0,24	0,24	0,22	0,2	0,18	0,16	0,14	0,12	0,16	0,19	0,23	0,26	0,27	0,27	0,27	0,27	0,27	0,25	0,2	0,17	0,15	0,12	0,07	0,01	-0,03	-0,07	-0,11
0,18	0,2	0,2	0,2	0,18	0,16	0,14	0,12	0,1	0,12	0,15	0,17	0,2	0,21	0,21	0,21	0,21	0,21	0,21	0,16	0,13	0,11	0,08	0,03	-0,01	-0,05	-0,09	-0,13
0,12	0,14	0,16	0,16	0,16	0,14	0,12	0,1	0,08	0,08	0,11	0,12	0,14	0,15	0,15	0,15	0,15	0,15	0,15	0,12	0,09	0,07	0,04	0,01	-0,03	-0,07	-0,11	-0,15
0,04	0,08	0,1	0,12	0,12	0,12	0,1	0,08	0,06	0,05	0,07	0,08	0,08	0,09	0,09	0,09	0,09	0,09	0,09	0,08	0,05	0,03	0,02	-0,01	-0,05	-0,09	-0,13	-0,17
-0,04	0	0,04	0,06	0,08	0,08	0,08	0,06	0,04	0,03	0,03	0,04	0,04	0,04	0,04	0,04	0,04	0,04	0,04	0,04	0,02	0,01	0	-0,03	-0,07	-0,11	-0,15	-0,19
-0,12	-0,08	-0,04	0	0,02	0,04	0,04	0,04	0,02	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0	0,01	0,01	0,01	0	-0,01	-0,04	-0,08	-0,12	-0,16	-0,2
-0,19	-0,15	-0,11	-0,07	-0,03	-0,01	0,01	0,01	0,01	0	0	0	-0,01	-0,01	-0,01	-0,01	-0,01	-0,01	0	0	0	0	-0,01	-0,04	-0,08	-0,12	-0,16	-0,2
-0,25	-0,21	-0,17	-0,13	-0,09	-0,05	-0,03	-0,01	-0,01	-0,01	-0,01	-0,01	-0,02	-0,04	-0,04	-0,02	-0,01	0	0	0,01	0,01	0	-0,02	-0,05	-0,09	-0,13	-0,17	-0,21
-0,31	-0,27	-0,23	-0,19	-0,15	-0,11	-0,07	-0,05	-0,04	-0,04	-0,04	-0,04	-0,05	-0,04	-0,01	0	0,01	0,01	0,01	0,02	0,03	0	-0,04	-0,07	-0,11	-0,15	-0,19	-0,23
-0,37	-0,33	-0,29	-0,25	-0,21	-0,17	-0,13	-0,1	-0,09	-0,08	-0,08	-0,08	-0,04	0	0,03	0,04	0,04	0,04	0,04	0,04	0,06	0	-0,04	-0,09	-0,13	-0,17	-0,21	-0,25
-0,44	-0,4	-0,36	-0,32	-0,28	-0,24	-0,2	-0,17	-0,16	-0,13	-0,07	-0,04	-0,01	0,01	0,04	0,05	0,08	0,08	0,08	0,08	0,08	0,02	-0,04	-0,09	-0,15	-0,19	-0,23	-0,27
-0,49	-0,47	-0,45	-0,41	-0,37	-0,31	-0,25	-0,18	-0,13	-0,08	-0,05	-0,02	-0,01	0,01	0,01	0,02	0,03	0,03	0,04	0,05	0,05	0,04	-0,02	-0,09	-0,15	-0,16	-0,12	-0,08
-0,35	-0,35	-0,33	-0,31	-0,27	-0,23	-0,17	-0,12	-0,07	-0,04	-0,01	-0,01	0	0	0	0	-0,01	-0,03	-0,04	-0,05	-0,05	-0,05	-0,05	-0,01	0,01	0,04	0,08	0,12
-0,21	-0,21	-0,21	-0,19	-0,17	-0,13	-0,09	-0,04	-0,01	0,02	0,01	0,01	0,01	0,01	-0,01	-0,02	-0,05	-0,07	-0,12	-0,15	-0,15	-0,15	-0,03	0,08	0,18	0,24	0,28	0,32
-0,07	-0,07	-0,07	-0,07	-0,05	-0,03	0,01	0,04	0,07	0,07	0,05	0,04	0,04	0,01	-0,01	-0,05	-0,08	-0,13	-0,18	-0,19	-0,13	-0,08	0	0,1	0,2	0,31	0,43	0,52
0,07	0,07	0,07	0,07	0,07	0,09	0,11	0,14	0,15	0,13	0,1	0,09	0,07	0,03	-0,03	-0,07	-0,12	-0,16	-0,16	-0,12	-0,08	-0,04	0,02	0,12	0,22	0,33	0,45	0,57
0,21	0,21	0,21	0,21	0,21	0,21	0,23	0,24	0,25	0,2	0,17	0,15	0,11	0,03	-0,03	-0,09	-0,08	-0,08	-0,09	-0,08	-0,04	0	0,05	0,14	0,24	0,35	0,47	0,59
0,35	0,35	0,33	0,31	0,29	0,27	0,25	0,25	0,26	0,29	0,24	0,18	0,12	0,05	-0,01	-0,03	-0,03	-0,03	-0,04	-0,04	-0,01	0,03	0,08	0,15	0,25	0,36	0,48	0,6
0,29	0,27	0,25	0,23	0,21	0,19	0,17	0,16	0,17	0,19	0,15	0,08	0,06	0,04	0,01	-0,02	-0,01	-0,01	-0,01	-0,01	0	0,04	0,09	0,16	0,25	0,35	0,47	0,59
0,22	0,2	0,18	0,16	0,14	0,12	0,1	0,09	0,1	0,12	0,06	0,06	0,04	0,03	0,01	0	-0,01	0	0	0,01	0,02	0,05	0,1	0,16	0,25	0,35	0,45	0,57
0,17	0,15	0,13	0,11	0,09	0,07	0,05	0,04	0,05	0,07	0,04	0,03	0,02	0,01	0,01	0	0	0	0	0,01	0,03	0,07	0,12	0,18	0,25	0,35	0,45	0,55
0,13	0,11	0,09	0,07	0,05	0,03	0,01	0,01	0,02	0,04	0,02	0,01	0,01	0	0	0	0	-0,01	0	0,01	0,03	0,06	0,13	0,2	0,26	0,35	0,45	0,55
0,09	0,07	0,05	0,03	0,01	-0,01	-0,01	-0,01	0	0,01	0,01	0	0	-0,01	-0,01	-0,01	0	-0,01	-0,02	0,01	0,03	0,05	0,11	0,19	0,27	0,35	0,45	0,55
0,04	0,02	0	-0,02	-0,04	-0,04	-0,04	-0,04	-0,02	-0,01	-0,01	-0,01	-0,01	-0,02	-0,03	-0,03	-0,02	-0,01	-0,03	-0,01	0,03	0,05	0,09	0,17	0,25	0,33	0,44	0,55
-0,02	-0,04	-0,06	-0,08	-0,08	-0,08	-0,08	-0,07	-0,04	-0,04	-0,04	-0,04	-0,03	-0,04	-0,06	-0,05	-0,04	-0,02	-0,03	-0,03	0,01	0,05	0,07	0,15	0,23	0,31	0,4	0,52
-0,08	-0,1	-0,12	-0,12	-0,12	-0,12	-0,12	-0,09	-0,08	-0,08	-0,08	-0,08	-0,07	-0,06	-0,08	-0,08	-0,06	-0,04	-0,03	-0,05	-0,01	0,03	0,07	0,13	0,21	0,29	0,37	0,48
-0,14	-0,16	-0,16	-0,16	-0,16	-0,16	-0,15	-0,12	-0,12	-0,12	-0,12	-0,12	-0,11	-0,09	-0,1	-0,12	-0,08	-0,06	-0,04	-0,05	-0,03	0,01	0,05	0,11	0,19	0,27	0,35	0,44

a



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	-0,01	-0,01	-0,01	-0,01	-0,01	-0,01	-0,01	-0,01	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	-0,01	-0,01	-0,02	-0,04	-0,04	-0,02	-0,01	-0,02	-0,04	-0,02	-0,01	0	0	0	0	0	0
0	0	0	0	0	0	0	0	-0,01	-0,01	-0,01	-0,02	-0,01	-0,01	-0,01	-0,01	-0,01	0	-0,01	-0,04	-0,04	0	0,01	0,01	0	0	0	0
0	0	0	0	0	0	0	0	-0,01	0	0	0	0,01	0,01	0,01	0,01	0,01	0,01	0	-0,03	-0,01	0	0,02	0,01	0	0	0	0
0	0	0	0	0	0	0	0	0,01	0,02	0,04	0,04	0,02	0,01	0,01	0,01	0,01	0	-0,01	-0,02	-0,01	0	0,01	0,01	0	0	0	0
0	0	0	0	0	0	0	0,01	0,02	0,01	0,01	0,02	0,01	0	0	0	-0,01	-0,02	-0,02	-0,01	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0,01	0,02	0,01	0	0	0,01	0	0	0	-0,01	-0,02	-0,01	-0,01	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0,01	0,04	0,02	0,01	0	0	0	0	-0,01	-0,02	-0,01	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0,01	0,01	0,01	0,01	0,01	0,01	0,01	-0,01	-0,02	-0,01	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0,01	0	-0,01	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	-0,01	-0,02	-0,01	0,01	0,01	0,01	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	-0,01	-0,02	-0,01	0	0	0,01	0,01	0,01	0,01	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	-0,01	-0,02	-0,01	0	0	0	0	0	0,01	0,01	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	-0,01	-0,02	-0,01	0	0	0	0	0	0	0	0,01	0,01	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	-0,01	-0,02	-0,01	0	0	0	0	0	0	0	0	0,01	0,01	0	0	0	0	-0,01	-0,01	0	0
0	0	0	0	0	0	-0,01	-0,02	-0,01	0	0	0	0	0	0	0	0	0,01	0,02	0,01	0	-0,01	-0,01	-0,01	-0,01	0	0	0
0	0	0	0	0	0	-0,01	-0,04	-0,01	0	0	0	0	0	0	0	0,01	0,01	0	-0,01	-0,01	-0,01	-0,01	0	0	0	0	0
0	0	0	0	0	0	-0,01	-0,02	-0,01	0	0	0	-0,01	-0,01	-0,01	0	0,01	0,01	0	-0,01	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	-0,01	-0,01	-0,01	-0,01	-0,01	-0,02	0	0	0,01	0	0,01	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0,01	0	-0,01	-0,01	0	0	0,02	0,01	0,01	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0,01	0,01	0,01	0,01	0,02	0,01	0,01	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0,01	0,01	0,01	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b

Fig. 44 Table of weights of the first layer for a neuron comparing the 2_172 and 5_102 samples, (a) for the part of the binary matrix, in which the light pixels of the image (> 150) correspond to 1, (b) for the part of the binary matrix, in which the darkened pixels of the image (<150) matches 1.

Paragraf 7.1. the task and its relevance are given.

Paragraf 7.2. a neural network is created based on metric recognition methods. For this, a software module was implemented in the Builder C ++ environment (fig. 42), which allows you to create and test a neural network of metric recognition methods based on a selected set of samples [28].

A selected set of 30 sample digits in the control sample of the MNIST database was used as the samples (Fig. 43).

```
Wh2 = -29
111111111111111111111111111111000000000000000000000000000000000000000000000000000000000000000000
Wh2 = -29
000000000000000000000000000000111111111111111111111111111111000000000000000000000000000000000000
Wh2 = -29
000000000000000000000000000000000000000000000000000000000000111111111111111111111111111111111110
Wh2 = -29
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001
Wh2 = -29
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
Wh2 = -29
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
```

a

```
Wh3 = 0
111000000000000000000000000000000
Wh3 = 0
000111000000000000000000000000000
Wh3 = 0
000000111000000000000000000000000
Wh3 = 0
000000000111000000000000000000000
Wh3 = 0
000000000000111000000000000000000
Wh3 = 0
000000000000000111000000000000000
Wh3 = 0
000000000000000000111000000000000
Wh3 = 0
000000000000000000000111000000000
Wh3 = 0
000000000000000000000000111000000
Wh3 = 0
000000000000000000000000000111000
Wh3 = 0
000000000000000000000000000000111
```

b

Fig. 45 (a) Fragment of weights for the second layer, (b) All weights of the third layer.

This paragraf also describes the procedure for calculating the values of weights for neurons in the first, second and third layers. The dimension of the weight tables of the first layer neuron is determined by the dimension of the MNIST images. An example of a calculated weight table is shown in fig. 44. In fig. 45*ab* also gives the values of weights and thresholds of connections of neurons in the second and third layers.

Also, this paragraf describes the corrections and changes made to transfer neurons of the network with a threshold activation

function into continuous function, and in particular for using the sigmoid activation function in the network [28]:

$$f(Sw) = \frac{1}{1+e^{-Sw}} \tag{57}$$

Paragraf 7.3. the results of recognition of the MNIST control base (10,000 images) in the resulting neural network without training the neural network are presented [27, 28]. Table 8 provides a summary and results separately for each recognizable digit in the MNIST control base. All the calculations described in this work were carried out on one computer in the software module shown in fig. 42, implemented in the Builder C ++ environment. For the entire process of creating a neural network and calculating all weights in the software module in fig. 42, the total spent time $t_{creating}$ = 0.5469 sec. was recorded, that is, less than a second.

Tab. 8 Recognition results of the MNIST control base (10,000 characters) without training

s0 = 834	i0 = 980	p0 = 85%
s1 = 968	i1 = 1135	p1 = 85%
s2 = 530	i2 = 1032	p2 = 51%
s3 = 454	i3 = 1010	p3 = 44%
s4 = 410	i4 = 982	p4 = 41%
s5 = 411	i5 = 892	p5 = 46%
s6 = 586	i6 = 958	p6 = 61%
s7 = 556	i7 = 1028	p7 = 54%
s8 = 773	i8 = 974	p8 = 79%
s9 = 750	i9 = 1009	p9 = 74%
Total: i = 10000, s = 6272, p = 62%		

Paragraf 7.4. describes the stochastic learning algorithm - back propagation, implemented in the software module in fig. 42. Also shown are the features of samples to neurons for the diagram in fig. 33 when implementing the *back propagation* algorithm for this network [28].

Paragraf 7.5. shows the results of comparing two performed experiments of the obtained neural network implemented in the software module in fig. 42 [27, 28]. In the first experiment, the resulting neural network was trained with the already calculated weight and threshold values. In the second experiment, based on the same neural network, the neural network was trained in the classical way - that is, without using the preliminary calculation of the values of the weights and thresholds.

Tab. 9 Comparison of the neural network training results for the MNIST training set (60,000 images) for each training epoch.

		training neural network with precomputed weights				training a neural network with random initialization of weights in the range [-0.5; 0.5]			
№ era	Learning rate	Qty. learn fig.	% recogn. fig.	S_{err}	time, min.	Qty. learn fig.	% recogn. fig.	S_{err}	time, min.
1	0,1	43932	73%	1199	159	35370	59%	1935	256
2	0,1	49748	83%	737	98	46033	76%	1051	139
3	0,02	52285	87%	545	72	49195	82%	784	104
Total training time, in minutes					329	Total training time, in minutes			499

Tab. 10 Comparison of neural network training results with validation on the MNIST control sample (10,000 images) for each training epoch.

№ era	Learning rate in the era	Training a neural network with precomputed weights	Training a neural network with random initialization of weights
1	0,1	9145	8894
2	0,1	9282	9116
3	0,02	9449	9256

For the second experiment, the values of the weights were determined randomly. The comparison results are shown in table 9 - for the MNIST training database - 60,000 images, and in Table 10 -

for the MNIST control database - 10,000 digit images. According to the results of the tables, it can be seen that for the case of training a neural network with calculated values of the weights, the results at all training epochs are better than for the classical training method.

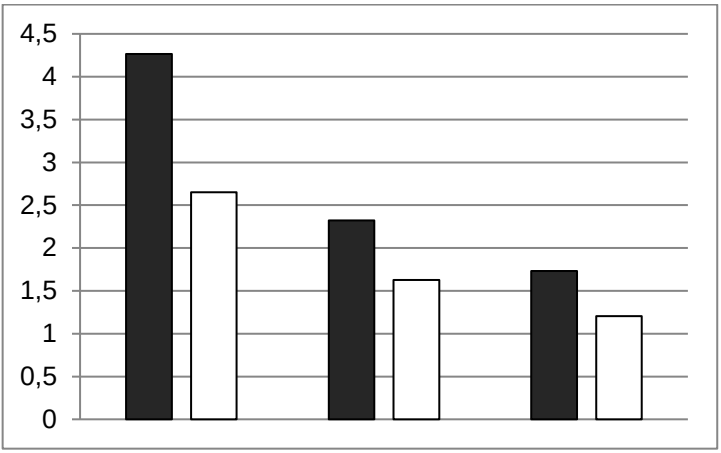


Fig. 46 Diagram of comparison of the elapsed time for each epoch for two experiments (white - with calculated values of the weights, black – not calculated values).

In addition, the time spent on training the neural network with the calculated weights is also less at all epochs compared to the neural network trained from zero (fig. 46). In general, 329 minutes = 5 hours 29 minutes were spent on training the neural network with the calculated weight values. When training the neural network from scratch, the total training time was 499 minutes. = 8 hours 19 minutes That is, the second experiment took almost 3 hours more time. These obtained data confirm the process of accelerating the procedure for creating and training a neural network with pre-calculated values of the weights and thresholds of the neural network.

Paragraf 7.6 presents the conclusions of the obtained experiments.

Chapter VIII discusses the possibility of almost instantaneous determination of the values of the neural network weights (without

calculation and training) based on the parameters of the electrostatic field [29, 30].

Paragraf 8.1. the set goal is described, solved in the 8th chapter.

Paragraf 8.2. it is proposed to use as a measure of proximity the formula of electrostatics - intensity [29]. It is shown how the values of the weights can be calculated on the basis of the intensity formula (58). The problematic aspects of this approach and the ways to resolve them are considered.

$$w_{i,j}^{(1)} = \overline{E}_1 - \overline{E}_2 = \frac{K \otimes q_1}{d_1^2} - \frac{K \otimes q_2}{d_2^2}, \qquad (58)$$

In this case, the values of the intensity of the electrostatic field are determined by the intensity sensors, the values from which are fed to the inputs of the neural network (Fig. 47).

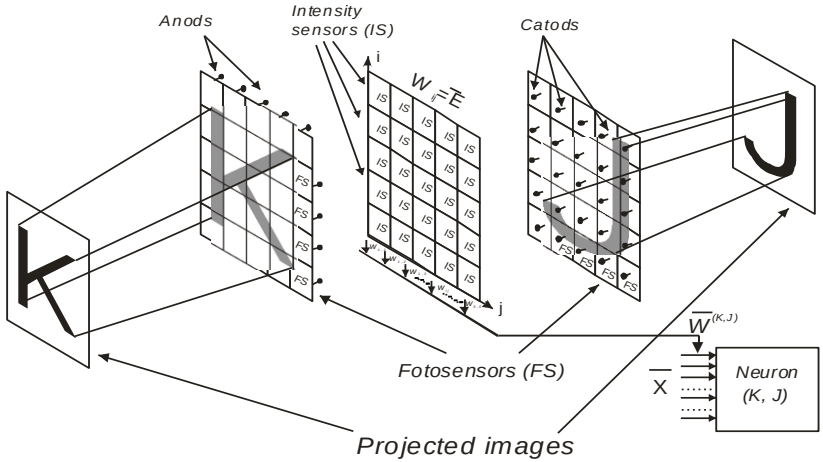


Fig. 47 Diagram showing obtaining the weights of a neuron of the first layer for projected images "*K, J*" using photosensors and intensity sensors.

Paragraf 8.3. it is also proposed to use as a measure of proximity the formula of electrostatics - potential [29]. It is shown how the value of the weights can be calculated on the basis of the (59) potential formula (Fig. 48, 49).

$$\varphi_{(i,j)}^{Sum} = \sum_{\substack{i=0,\\ j=0}}^{\substack{i=28,\\ j=28}} \varphi_{(i,j)}^{(Img1)} - \sum_{\substack{i=0,\\ j=0}}^{\substack{i=28,\\ j=28}} \varphi_{(i,j)}^{(Img2)} = \sum_{\substack{i=0,\\ j=0}}^{\substack{i=28,\\ j=28}} \frac{K*q_{(i,j)}^{(Img1)}}{r_{(i,j)}^{(Img1)}} -$$

$$- \sum_{\substack{i=0,\\ j=0}}^{\substack{i=28,\\ j=28}} \frac{K*q_{(i,j)}^{(Img2)}}{r_{(i,j)}^{(Img1)}}, \qquad (59)$$

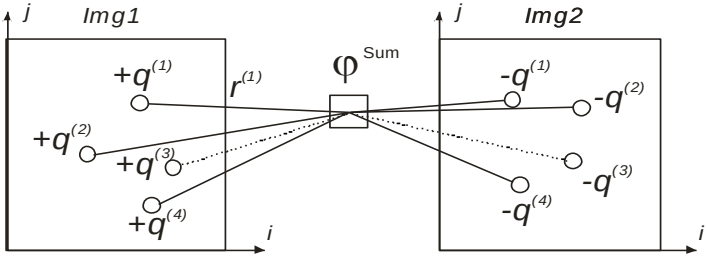


Fig. 48 Scheme for determining the potential-weight in one cell of the weight table (the plane of the potentiometer sensors) of the first layer.

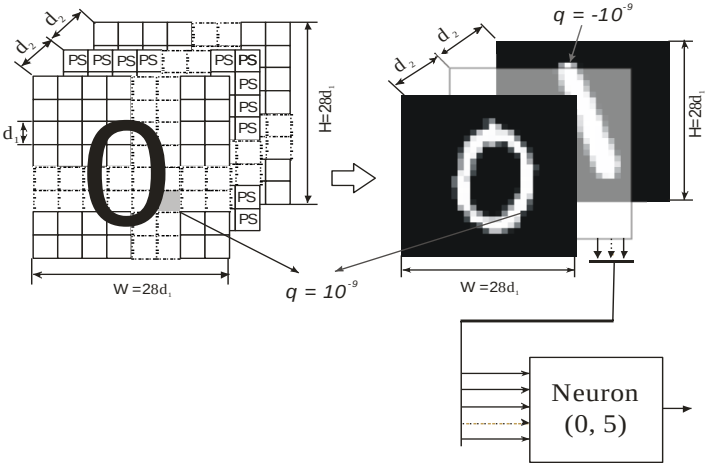


Fig. 49 Diagram of the simulated model of charged image planes 0_157 and 1_46 from the MNIST base and the plane of the potentiometer sensors (ps), which determines the neuron of the first layer of the neural network.

Tab. 11 Tables of weights of the neuron of the first layer performing comparison of images 0_157, 1_46 of the MNIST base. The values of the neural network weights are determined by the total potential.

k = 0, *k1* = 5, 0_157, 1_46 *Wh1* = -26241															
1	2	3	4	5	6	7	8	9	10	11	12	13	…..	27	28
218	217	216	204	192	182	160	139	113	98	101	129	188		395	378
227	226	221	212	195	175	146	105	58	16	-2	33	122		424	401
241	240	232	221	202	171	130	69	-19	-116	-188	-162	-14		456	427
256	255	248	237	215	178	124	36	-108	-316	-529	-544	-284		486	447
274	275	274	259	238	199	139	28	-172	-530	-986	-1063	-638		519	474
295	300	297	291	275	243	184	70	-145	-556	-1095	-1169	-626		549	501
318	327	333	331	323	305	263	180	8	-319	-754	-792	-267		578	530
342	358	373	376	386	389	375	343	272	146	-26	-157	-18		610	551
372	390	412	438	461	487	513	542	582	655	673	283	-81		639	573
396	426	458	498	538	592	663	758	889	1072	1119	492	-350		663	591
429	464	503	558	616	699	818	978	1209	1469	1436	597	-711		680	605
456	499	549	615	695	803	963	1192	1539	1911	1668	545	-893		693	610
480	528	592	665	763	898	1088	1368	1789	2201	1857	642	-782		693	609
504	557	625	709	823	977	1187	1500	1959	2380	2018	793	-646		683	604
519	579	653	745	868	1039	1264	1601	2079	2519	2166	953	-484		666	590
534	597	675	770	896	1075	1319	1666	2166	2632	2312	1148	-228		639	566
543	609	687	787	916	1095	1345	1702	2223	2721	2457	1382	151		605	541
545	610	692	791	916	1092	1345	1701	2229	2769	2580	1627	553		563	506
544	611	685	781	902	1070	1309	1650	2164	2734	2648	1873	991		521	472
536	600	673	761	878	1030	1238	1544	1986	2505	2584	2156	1538		474	440
527	584	656	734	836	970	1148	1385	1700	2052	2311	2440	2182		430	409
514	568	628	698	791	901	1042	1221	1439	1693	2008	2370	2441		391	382
498	544	600	663	741	832	942	1076	1233	1417	1655	1919	2125		359	353
481	521	570	625	686	763	850	951	1063	1197	1347	1508	1676		334	328
463	496	537	589	639	695	767	841	924	1014	1109	1211	1300		315	314
439	472	505	549	592	639	691	748	805	866	927	985	1025		300	296
419	447	480	512	545	590	625	669	705	748	788	817	832		289	286
398	423	449	477	508	541	568	598	627	654	679	694	695		278	275

In this case, the values of the potential of the electrostatic field are determined by the sensors potentiometers, the values from which are fed to the inputs of the neural network (Fig. 49).

Paragraf 8.4. experiments are carried out to determine the values of the neural network weights based on a simulated electrostatic field [30]. The experiments were carried out in the Builder C ++ software environment.

Paragraf 8.5. the results of experiments are checked [30]. The results showed that when using the electrostatic field potentials as the weights (Tab. 11), the number of correctly recognized elements of the MNIST control base is more than 50% on 30 samples. And this number increases with an increase in the number of samples used (Tab. 12), and also changes with a change in the physical indicators of the potential of the electrostatic field, such as the values of point charges and the distances between the panels of the projected images.

Tab. 12 Test results on the MNIST test base using 50 samples for the diaqram in fig. 49, where the point charge $q = 10^{-9}$ Cl. , the distance between point charges and charged panels is $d_2 = 4$ cm, $d_1 = 2$ cm.

s0 = 852	i0 = 980	p0 = 87%
s1 = 1123	i1 = 1135	p1 = 99%
s2 = 516	i2 = 1032	p2 = 50%
s3 = 585	i3 = 1010	p3 = 58%
s4 = 775	i4 = 982	p4 = 79%
s5 = 588	i5 = 892	p5 = 66%
s6 = 555	i6 = 958	p6 = 58%
s7 = 616	i7 = 1028	p7 = 60%
s8 = 633	i8 = 974	p8 = 65%
s9 = 726	i9 = 1009	p9 = 72%
10000, 6969, 70%		

Paragraf 8.6. an valuation of the results of this chapter is given.

In the final conclusions of the dissertation work, the main results and conclusions based on the research of the dissertation work are given. The scientific novelty and practical value of the work are also given.

In the Appendix of the dissertation work, the developed program codes in the Builder C ++ environment are given, in which a neural network is implemented based on metric recognition methods, all weights are calculated, the weights are determined based on the parameters of the electrostatic field and in which the comparative experiments and estimates given in the work were performed.

In addition, the appendix contains tables of weights for the zero and first layers, as well as the values of all outputs (functions of states and activations of the neural network) of the implemented neural network for selected input recognizable elements of the neural network from the MNIST database.

KEY RESULTS OF WORK

1. New architectures of feedforward neural networks with calculated parameters are proposed.
2. The mathematical models of the proposed architectures of neural networks are presented.
3. Methods for minimizing the number of neurons and connections of the proposed neural networks are proposed.
4. Analytical expressions are proposed that calculate the parameters of the structure of the neural network: the number of neurons, layers, connections.
5. Analytical expressions are proposed that allow calculating the values of the weight and threshold values of the connections of neurons.
6. The schemes and algorithms for the application of the proposed neural networks in multitasking applications have been developed.
7. Algorithms and schemes for applying the proposed neural networks in problems with fuzzy conclusions have been developed.
8. The conditions determining the distribution of weight and threshold values of the multilayer perceptron are found and presented.
9. On the basis of the Widrow-Hoff learning algorithm, it is

shown and mathematically substantiated the tendency of learning algorithms for neural networks of direct propagation to the distribution conditions of the weight values obtained for the proposed computable neural networks.

10. A general algorithm for calculating the weight and threshold values of neurons in the second and third layers of a multilayer perceptron based on metric recognition methods is proposed.
11. On the basis of the architectures of neural networks of metric methods of recognition, the architectures of neural networks that implement pairwise sequential separation of images are proposed.
12. Implemented a software module in the Builder C ++ software environment, which implements neural networks based on metric recognition methods.
13. Based on the MNIST numbers, it is shown that in a fraction of a second, a neural network is created and all the weight values of the weight tables are calculated.
14. It has been shown that different activation functions are applicable to neurons of these architectures, including continuous activation functions such as sigmoid activation functions.
15. In practice, it was also shown that this neural network can also be retrained by the back propagation algorithm. For this, a stochastic algorithm - back propagation was implemented and shown, in relation to the proposed architecture of the neural network.
16. On the basis of MNIST, the acceleration of the procedure for creating and training a neural network with calculated values of weights has been experimentally proved in comparison with the classical approach.
17. Shown theoretically and experimentally the possibility of almost instantaneous determination of the values of the weights without training and calculating the weight values, based on such parameters of the electrostatic field as the intensity and potential.

The main content of the dissertation work is set out in the following publications:

1. Гейдаров, П.Ш. Структурно признаковый метод распознавания рукопечатных символов // Известия НАН Азерб., серия физико математических и технических наук, — 2006. Т. XXV, №3, — с. 65-70.

2. Нусратов, О.К., Гейдаров, П.Ш. Позиционно-бинарный метод распознавания рукопечатных символов // Труды VII Международного симпозиума «Интеллектуальные системы», INTELS-2006, — Краснодар, Россия, — 2006, — с. 474-477.

3. Нусратов, О.К, Гейдаров, П.Ш. Метод распознавания рукопечатных символов и текстов // Труды Международной Конференции PCI, 2006 «Проблемы Кибернетики и информатики», — Баку: — 2006. — с. 52-58.

4. Нусратов, О.К. Метод распознавания рукопечатных символов, Решение государственного агенства по стандартизации, метрологии и патента Азербайджанской Республики,02.10.2006 A 2006 0182 / Гейдаров П.Ш. — 2006.

5. Гейдаров, П.Ш. Алгоритмы устранения ошибочных сегментаций в структурно-сегментном представлении символа // Известия НАН Азерб., серия физико математических и технических наук, — 2008. Т. XXVIII, №3, — с. 115-119.

6. Nusratov, O.Q. Position-Width-Pulse Algorithm for Recognition of Handwritten Characters / P.Sh. Geidarov // Automatic Control And Computer Sciences, — 2007. V. 41, №6, — pp.332-336. **(Springer, Scopus, Web of Science: ESCI)**

7. Geidarov, P.Sh. Algorihm determination of spacing interval up to object // The second international conference “Problem of Cybernetics and informaties.” — V. 2, 2008. — p. 68-71.

8. Geidarov, P.Sh. Neural networks on the basis of the sample method // Automatic Control And Computer Sciences, —

2009. V. 43, № 4, — p. 203-210. **(Springer, Scopus, Web of Science: ESCI)**

9. Гейдаров, П. Ш. Нейронные сети на основе метрических методов распознавания в применении к задачам с нечеткими выводами // Искусственный интеллект и принятия решений, — 2010. № 2, — с. 77-88. **(Web of Science : RSCI)**

10. Гейдаров, П.Ш. О возможностях электронного научного семинара // Социологические исследования, — 2010. № 8, — с. 135–137. **(Web of Science : SSCI)**

11. Гейдаров, П.Ш. Алгоритм определения расположения и размеров объекта на основе анализа изображений объектов // Компьютерная оптика, — Саратов: — 2011. Т. 35, №2, — с. 275-280. **(Scopus, Web of Science: ESCI)**

12. Гейдаров, П.Ш. Автоматизация процедуры обращений граждан в госучреждения и электронное правительство // Социологические исследования, — 2012. № 11. — с. 71-80. **(Web of Science : SSCI)**

13. Гейдаров, П.Ш. Электронный научный семинар // Elektron elm problemləri üzrə 1 respublika elmi-praktiki konfransın materialları, — Bakı: — 2012. — с. 54-57.

14. Geidarov, P.Sh. Multitasking application of neural networks implementing metric methods of recognition // Autom. Remote Control, — 2013. V. 74, № 9, — p. 1474–1485. (**Scopus, Web of Science: SCI Exp.)**

15. Geidarov, P.Sh. Clearly Defined Neural Networks Architecture // Optical Memory & Neural Networks, — 2015. V. 24, I. 3, — p. 209-219. **(Springer, Scopus)**

16. Гейдаров, П. Ш. Балловая система оценки научных трудов и электронный научный семинар // Социологические исследования, — 2015. № 4. — с. 162-167. **(Web of Science : SSCI)**

17. Гейдаров П.Ш. Алгоритм кратчайшего маршрута на основе выделенного набора маршрутов // Информационные технологии, — 2016. Т. 22, №9, — с. 660–668. **(Web of Science : RSCI)**

18. Geidarov, P. Sh. Clearly defined architectures of neural networks and multilayer perceptron // Optical Memory and Neural Networks, — 2017. V. 26, — p. 62–76. **(Springer, Scopus)**

19. Гейдаров, П.Ш. Алгоритм реализации метода ближайшего соседа в многослойном персептроне // Труды СПИИРАН, — 2017. Т. 51, — с. 123-151. **(Scopus)**

20. Гейдаров, П.Ш. Перспективы создания государственной системы мобильного видеомониторинга // Информационное общество, — 2017. №4-5. — с. 114-121. **(Web of Science : RSCI)**

21. Гейдаров, П.Ш. Нейронные сети прямого распространения с вычисляемыми параметрами // Информационные технологии, — 2017. Т. 23, № 7, — с. 543-552. **(Web of Science : RSCI)**

22. Гейдаров, П.Ш. Архитектура нейронной сети с попарно последовательным разделением образов. // Прикладная дискретная математика, — 2018. №41, — с. 98-109. **(Scopus, Web of Science : ESCI)**

23. Geidarov, P.Sh. Neural Networks with Image Recognition by Pairs // Optical Memory and Neural Networks, — 2018. V. 27, № 2, — p. 113–119. **(Springer, Scopus)**

24. Гейдаров, П.Ш. Автоматизированный выбор состава научного совета электронного научного семинара с использованием нейронных сетей на основе метрических методов распознавания // Международная научно-практическая конференция “Научно-практические исследования”, — Россия, Омск: — 2020. № 1-4, Т. 24, — с. 35-42.

25. Гейдаров, П.Ш. Нейронные сети на основе метрических методов распознавания // XXVI Международная научно-практическая конференция «Advances in Science and Technology», — Россия, Москва: — 2020. Часть I, – с. 77-81.

26. Гейдаров, П.Ш. Алгоритм вычисления значений весов синапсов первого слоя нейронной сети на основе метрических методов распознавания. Часть 1. // Информационно-

Управляющие Системы, — Санкт-Петербург: — 2020. №2, — с. 20-30. **(Scopus)**

27. Гейдаров, П.Ш. Алгоритм вычисления значений весов синапсов первого слоя нейронной сети на основе метрических методов распознавания. Часть 2. // Информационно-Управляющие Системы, — Санкт-Петербург: — 2020. №3, — с. 25-38. **(Scopus)**

28. Geidarov, P. Sh. Comparative Analysis of the Results of Training a Neural Network with Calculated Weights and with Random Generation of the Weights // Automation and Remote Control, NewYork, — 2020. V. 81, №. 7, — с. 1211-1229. **(Scopus, Web of Science : SCI Exp.)**

29. Гейдаров, П.Ш. О возможности определения значений весов нейронной сети электростатическим полем // Искусственный интеллект и принятие решений, — Москва: – 2021. №1, – с. 33-49. **(Web of Science : RSCI)**

30. Гейдаров, П.Ш. Эксперимент по созданию нейронной сети с весами, определяемыми потенциалом имитированного электростатического поля // Искусственный интеллект и принятие решений, — Москва: – 2021. №2. – с. 78-92. **(Web of Science : RSCI)**

The defense will be held on 26 november 2021 at 14^{00} at the meeting of the Dissertation council ED 1.20 of Supreme Attestation Commission under the President of the Republic of Azerbaijan operating at Institute of Control Systems of Azerbaijan National Academy of Sciences.

Address: Az 1141, Baku, st. B. Vahabzade, 9

Dissertation is accessible at the Library of the Institute of Control Systems of Azerbaijan National Academy of Sciences.

Electronic versions of dissertation and its abstract are available on the official website of the Institute of Control Systems of Azerbaijan National Academy of Sciences.

Abstract was sent to the required addresses on 21 october 2021.

Signed for print: 07.10.2021

Paper format: A5

Volume: 90381

Number of hard copies: 20