Neural Networks 28 (2012) 15–23

Contents lists available at SciVerse ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Investigation of efficient features for image recognition by neural networks

Alexander Goltsev *, Vladimir Gritsenko

International Research and Training Center of Informational Technologies and Systems of National Academy of Sciences of Ukraine, Pr. Glushkova 40, Kiev 03680, Ukraine

ARTICLE INFO

Article history:
Received 16 September 2011
Revised and accepted 13 December 2011

Keywords:
Feature
Neuron
Connection
Training
Recognition
Classifier

ABSTRACT

In the paper, effective and simple features for image recognition (named LiRA-features) are investigated in the task of handwritten digit recognition. Two neural network classifiers are considered—a modified 3-layer perceptron LiRA and a modular assembly neural network. A method of feature selection is proposed that analyses connection weights formed in the preliminary learning process of a neural network classifier. In the experiments using the MNIST database of handwritten digits, the feature selection procedure allows reduction of feature number (from 60 000 to 7000) preserving comparable recognition capability while accelerating computations. Experimental comparison between the LiRA perceptron and the modular assembly neural network is accomplished, which shows that recognition capability of the modular assembly neural network is somewhat better.

1. Introduction

Very effective and very simple recognition features have been proposed in the following papers: Kussul, Kasatkina, and Lukovich (1999); Kussul, Baidyk, Kasatkina, and Lukovich (2001); Kussul and Baidyk (2002, 2003, 2006); Kussul, Baidyk, and Wunsch (2010). Effectiveness of these features is demonstrated in such different tasks as recognition of handwritten digits of the MNIST database (Kussul & Baidyk, 2002, 2003, 2006; Kussul et al., 2001, 2010, 1999), micromechanical control based on vision (Baidyk et al., 2004; Baidyk, Kussul, Makeyev, & Vega, 2008; Baidyk, Kussul, Makeyev, & Velasco, 2009; Kussul et al., 2010; Makeyev, Sazonov, Baidyk, & Martin, 2008; Martin-Gonzalez, Baidyk, Kussul, & Makeyev, 2010; Vega, Baidyk, Kussul, & Pérez Silva, 2011), and speaker identification (Kasatkina, Lukovych, & Pilipenko, 2006). In the experiments with handwritten digit recognition, a very high recognition quality is achieved—in particular, the classifier commits only 55 errors recognizing 10 000 test samples of the MNIST database (Kussul et al., 2010). This result exceeds the rates of many traditional and modern classifiers such as SVM (Dong, Krzyzak, & Suen, 2003; Vapnik, 1998), *k*-nearest neighbor classifier (Cover & Hart, 1967), perceptrons (Rosenblatt, 1962), Boosted LeNet and others (LeCun, 1998; LeCun, Bottou, Bengio, & Haffner, 1998). The mentioned top results (Kussul et al., 2010) were obtained using a simple 3-layer linear perceptron with a single layer of modifiable connections and a modified learning rule that do not require sophisticated mathematics and complex information processing. So, there are good reasons for further study of capabilities of these features. In the text below, the features are named LiRA-features.

In the present paper, two models of neural networks are employed in the experiments on handwritten digit recognition of the MNIST database. The first neural network is the LiRA (Limited Receptive Area) neural classifier presented in Kussul et al. (1999) and developed in Baidyk et al. (2004), Baidyk et al. (2008), Baidyk et al. (2009), Kussul and Baidyk (2002, 2003, 2006) Kussul et al. (2001), Kussul et al. (2010), Makeyev et al. (2008), Martin-Gonzalez et al. (2010), and Vega et al. (2011). A systematic experimental study of the LiRA classifier is also reported by Misuno, Rachkovskij, and Slipchenko (2005). LiRA is a modified version of 3-layer perceptron of Rosenblatt (1962). The second neural network is a modular assembly neural network proposed by Goltsev (1991) and developed in Goltsev (1996, 2004, 2005), Goltsev and Wunsch (1998, 2004), Goltsev, Kussul, and Baidyk (2004), Goltsev, Húsek, and Frolov (2005), Goltsev and Rachkovskij (2005), and Goltsev and Gritsenko (2009). Although both neural networks have a rather different structure, they use rather similar algorithms.

The well-known MNIST database (LeCun, 1998; LeCun et al., 1998) used in the present work is often employed for evaluation of different classifiers. The database contains 60 000 handwritten Arabic digits in its training set and 10 000 ones in the test set. Every black-and-white image is centered and placed in a raster of 28×28 pixels with 255 gray levels each. In the present work, these gray-level images are transformed into binary images according to a simple thresholding algorithm (Kussul et al., 2010, 1999; Misuno et al., 2005), so that a digit is depicted in the raster by one-valued pixels against the background of zero-valued pixels. In all experiments presented below, only this type of binary images is used.

* Corresponding author. Tel.: +380 44 497 37 75; fax: +380 44 502 25 49.
E-mail address: agoltsev@adg.kiev.ua (A. Goltsev).

0893-6080/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2011.12.002

2. The LiRA-features

Successful solutions of recognition tasks by simple linear perceptron (see Baidyk et al. (2004, 2008, 2009); Kussul et al. (1999, 2001, 2010); Kussul and Baidyk (2002, 2003, 2006); Makeyev et al. (2008); Martin-Gonzalez et al. (2010); Vega et al. (2011)) are based on a large amount (tens and hundreds of thousands) of simple LiRA-features. Let us designate the number of LiRA-features used by a classifier as N. For binary images, every LiRA-feature is a set of small number of pixels (4–10), some of the pixels are considered belonging to an object (one-valued pixels) and the others are considered belonging to background (zero-valued pixels). Let us designate a total number of feature pixels as H, the number of one-valued pixels as H^{pos}, and the number of zero-valued pixels as H^{neg}, $H = H^{\text{pos}} + H^{\text{neg}}$.$H$, H^{pos}, H^{neg} are the same for all N LiRA-features (e.g. $H = 6$, $H^{\text{pos}} = 2$, $H^{\text{neg}} = 4$).

The following algorithm of LiRA-feature formation is described in the referred publications. A $G \times G$ square is randomly located within a raster (for the MNIST database $G < 28$, e.g. $G = 9$). Coordinates of all H pixels of some feature are first randomly chosen within the square and then memorized together with the information whether they are one-valued or zero-valued. The same procedure is repeated to generate all N features.

This random feature formation algorithm is in conformity with such a modern approach in the field of neural networks as using random components (random patches, random weights) in recognition procedures (e.g. Coates, Lee, & Ng, 2011; Saxe et al., 2011).

When some image is presented at the input, all N features are sought in it. A feature is considered present in the image (and set to 1), if all one-valued and zero-valued pixels of the feature coincide with one-valued and zero-valued pixels of the image, respectively.

In the present work, the feature generation algorithm is slightly modified. As mentioned above, the MNIST database contains 60 000 samples of digits in its training set. We generate 60 000 LiRA-features using 60 000 training samples as follows. For the n-th feature, we generate its randomly placed $G \times G$ square and $H = H^{\text{pos}} + H^{\text{neg}}$ feature pixels inside it. Then we check if this n-th feature is present in the n-th training sample. If the feature is present, it is included in the feature set, otherwise the feature is regenerated until it is present in the n-th training sample. Thus, each feature from the feature set is present at least in one of the images. The idea of this modified algorithm is to create such a set of features that each of them is a randomly reduced image of at least one training sample.

3. The LiRA neural network classifier

The LiRA classifier consists of three layers of neurons: S, A, and R. An input image $\mathbf{I}$ is presented to the input layer S, which is a two-dimensional matrix (raster). The intermediate layer A comprises N neurons each of them represents single LiRA-feature. Let us designate the vectors of the input signals of the layers S and A as $\mathbf{s}$ and $\mathbf{a}$, and the vectors of the output signals of those layers as $\mathbf{S}$ and $\mathbf{A}$, respectively. The output layer R represents classes of images; let us designate the number of image classes as M. Fig. 1 schematically depicts the LiRA neural network classifier.

In the present work, the input layer S transforms zero-valued input signals of background image pixels into (-1) values, so that $\mathbf{s}_i \in \{0, +1\}$, and $\mathbf{S}_i \in \{-1, +1\}$. This transformation may be expressed as follows: $\mathbf{s} = \mathbf{I}$; $\mathbf{S}_i = 2\mathbf{s}_i - 1$. Let us designate the number of the S-layer neurons (S-neurons) as S. The output activity of S-neurons is transmitted to the neurons of the intermediate layer A by fixed, non-modifiable connections. Let us designate them by matrix $\mathbf{V}$: $V_{ij} \in \{-1, 0, +1\}$; $i = 1, 2, \ldots, S, j = 1, 2, \ldots, N$.

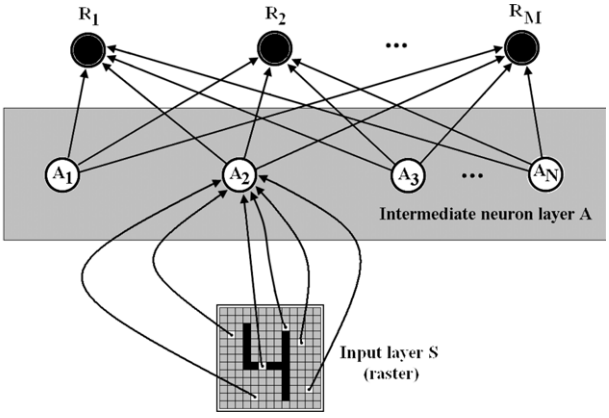

Fig. 1. The LiRA neural network classifier.

Total of H connections come to the input of each A-neuron from the S-layer neurons: total of H^{pos} connections with $(+1)$ weights and total of H^{neg} connections with (-1) weights; $H = H^{\text{pos}} + H^{\text{neg}}$. Nonzero connections (entries of $\mathbf{V}$) go to some A_j-th neuron from those S-neurons ("receptive pixels") that are chosen according to the random feature generation algorithm described in the previous section. The input of A_j-th neuron is determined as follows

$$a_j = \sum_{i=1}^{S} S_i V_{i,j}. \tag{1}$$

The output of the A_j-th neuron becomes one-valued ($A_j = 1$) if a_j is equal to H, and zero-valued ($A_j = 0$) otherwise. Thus, an A-neuron is activated only if all its H^{pos} positive connections come from the object pixels and all its H^{neg} negative connections come from background pixels of the input image.

The output of each A-neuron is transmitted to the inputs of all neurons of the R-layer (R-neurons) by means of M modifiable connections. Let us introduce a matrix $\mathbf{W}$ to denote connection weights. Then, an element $W_{jm}, j = 1, 2, \ldots, N$ of this matrix denotes the weight of connection which goes from the output of the A_j-th neuron to the input of the R_m-th neuron representing class m. Initially, all components of the matrix $\mathbf{W}$ are set to zero. During the training process, connection weights of the matrix $\mathbf{W}$ are modified as explained below. After training, LiRA is ready for recognition.

In the process of recognition, some test image is presented to LiRA. LiRA-features are detected in the image by means of the fixed connection structure $\mathbf{V}$ as explained above, and some pattern of the output activity (binary vector $\mathbf{A}$) is formed. This activity goes through the trained connection matrix $\mathbf{W}$ and is summarized by R-neurons. The activity of the R_m-th neuron is determined as

$$R_m = \sum_{j=1}^{N} A_j W_{j,m}. \tag{2}$$

In order to classify the input test image, the z-th R-neuron with maximum activity is found:

$$z = \arg \underset{m=1}{\overset{M}{\text{MAX}}} R_m. \tag{3}$$

The LiRA learning process uses a modified perceptron training rule (Kussul & Baidyk, 2002, 2003, 2006; Kussul et al., 2001, 2010, 1999). All training samples are presented to the S-layer in turn. Let us consider the input training sample which belongs to the class m. After recognition, the vector $\mathbf{R}$ is formed with its component R_m representing the correct class m. This actual activity of the R_m-th neuron is artificially decreased as R_m^*: $R_m^* = R_m(1 - T)$, where

T is the so-called defense space parameter, $0 \leq T < 1$. The decreased activity level R_m^* is compared to the activity levels of all other R-neurons to determine the winner according to Eq. (3). The training sample is considered correctly recognized if the R_m^* value exceeds the activity levels of all other R-neurons. In this case, no connection weight modification is performed. However, if some other R_d-neuron has the maximal activity, then the connection weights are modified as follows:

$$\mathbf{W}_{j,m} = \mathbf{W}_{j,m} + A_j \Delta W, \tag{4}$$

$$\mathbf{W}_{j,d} = \mathbf{W}_{j,d} - A_j \Delta W, \tag{5}$$

where $j = 1, 2, 3, \ldots, N$, and ΔW is a weight increment value (often $\Delta W = 1$). According to Eqs. (4) and (5), the connection weights may obtain positive or negative values.

Then, the next training sample is processed. A concept of epoch, which is often used in the field of learning algorithms, is useful for further explanation. The epoch is one complete cycle of successive consideration of all samples of the training set. A successive consideration of all training samples continues many times, epoch by epoch, until convergence, that is, absence of erroneous classification of any training sample.

It is well known (Rosenblatt, 1962) that a perceptron converges if a separating hyperplane exists in the A space, and if the perceptron is able to form a complex enough separating surfaces in the input space S to separate the training set. Introduction of the defense space has the aim to shift the separating hyperplane from the points of the training set analogous to the margin in the SVM (Cristianini & Shawe-Taylor, 2000; Vapnik, 1998) or AdaBoost (Freund & Schapire, 1999) paradigms. Therefore, introduction of the defense space generally results in improvement of generalization capability and classification quality. The LiRA training algorithm allows processing of large training sets in the space of LiRA-features of large dimensionality N and does not demand solving the optimization quadratic programming task, unlike SVM.

4. MNIST recognition using the LiRA classifier

In order to assess the optimal size of LiRA-features, in the first series of experiments, dependence of recognition capability of the LiRA classifier upon changing of total number of feature pixels H is studied under condition that the number of positive and negative feature pixels is equal, that is, $H^{\text{pos}} = H^{\text{neg}}$. For each experiment of this series, 60 000 LiRA-features are generated as explained in Section 2. Those features are extracted from all the training samples of the MNIST database. Then the LiRA training procedure of Section 3 with the value $T = 0.1$ (an intermediate value of T that provides convergence) is accomplished. After the classifier converges, its recognition capability is tested using 10 000 samples of the MNIST test set. The experimental results are presented in Fig. 2. The number indicating the total amount of epochs used for the classifier convergence is placed near each point of the plot.

These experiments demonstrate that the LiRA error rate strongly depends upon the feature size H. Also, the feature configuration, that is, the ratio between H^{pos} and H^{neg}, strongly influences the recognition rate, as shown in Kussul et al. (2010) and Misuno et al. (2005). So, the task of finding appropriate LiRA-feature size and configuration is not trivial and needs additional investigations. However, in the present work we chose the following feature parameters: $H = 6, H^{\text{pos}} = 2, H^{\text{neg}} = 4$ that provided rather good recognition rate in numerous experiments conducted in Misuno et al. (2005). These feature parameters were used in all the experiments described below.

Let us note that this series of experiments begins from the feature size $H = 10$ because preliminary experiments showed that

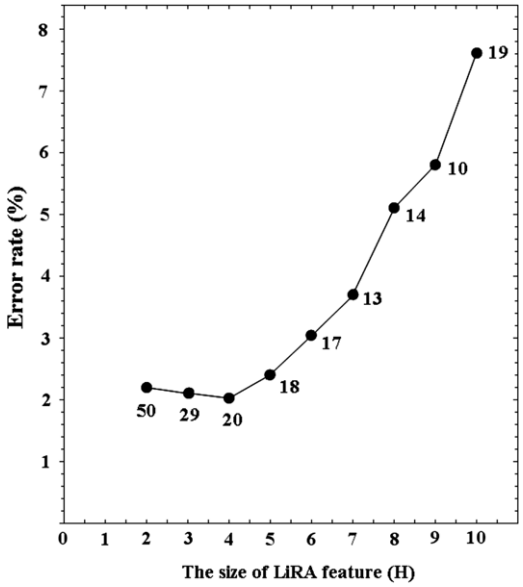

Fig. 2. Dependence of the MNIST base test set LiRA classifier error rate (%) upon the total number of feature pixels H. The number of epochs till the classifier convergence is placed near each plot point.

$H = 10$ is maximally possible feature size which allows detecting at least one feature in all test samples of the MNIST database.

Fig. 2 shows that the graphic has minimum of 2.04% (204 errors) at the feature size $H = 4$. When the parameter H increases, LiRA-features become too specific and, consequently, only a small number of them may be found in training and test samples. For recognition based upon a small number of features, even very specific, the error probability grows and the classifier recognition capability decreases, as seen in Fig. 2.

For the second experimental series, 60 000 new LiRA-features were generated with $H = 6, H^{\text{pos}} = 2, H^{\text{neg}} = 4$. Fig. 3 (lower curve) shows dependence of LiRA recognition capability on parameter T (the defense space). As in Fig. 2, the total number of epochs before convergence is placed near each point of the plot. If convergence is not achieved in 700 epochs, the training process is considered as unsuccessful and is terminated. The plot demonstrates that increasing defense space decreases the LiRA error rate. The best error rate of LiRA with 60 000 features is 1.5% (150 errors), what may be considered as good result for the classifier that does not use distortions of the training samples, as in Kussul et al. (2010), see also LeCun (1998).

5. Reduction of the feature pool

The classifiers that use a large amount of features are usually slow. For acceleration, the number of features may be reduced by means of feature selection procedure. Methods of feature selection may be categorized into two groups: the filter model that does not take the learning results into account, and the wrapper model that takes the learning results into account (Koller & Sahami, 1996). The wrapper model is often very expensive to run and can be intractable for a large number of features because it demands numerous iterations of the training process to estimate the selection quality. Filter methods are often based on the ideas from Information Theory (Cover & Thomas, 1991; Yang & Pedersen, 1997). In particular, dependence of the LiRA classifier performance on feature selection using the filter model is studied in Misuno et al. (2005), wherein the feature selection is performed using the criterion of mean mutual information between features and classes. It is reported in Misuno et al. (2005) that the feature

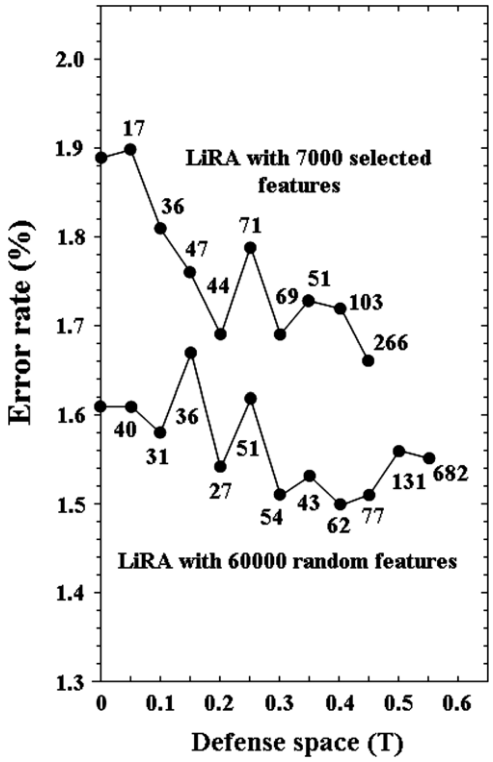

Fig. 3. Dependence of the MNIST base test set error rate (%) of the LiRA classifier upon the defense space size (parameter T) for LiRA with 7000 selected features (the upper curve) and 60 000 random features (the lower curve).

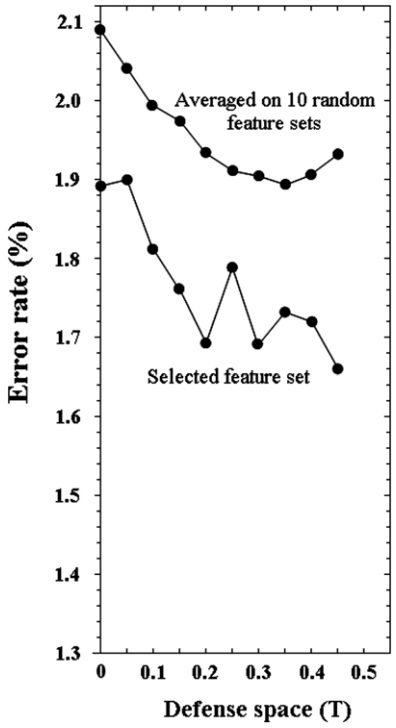

Fig. 4. Dependence of the LiRA classifier error rate (%) upon the defense space size (parameter T). The lower curve shows the results for 7000 selected features and the upper curve shows the mean results over 10 sets of 7000 random features.

selection procedure accelerates LiRA functioning in 20–200 times with retention of a comparable classification quality.

In this study we use a feature selection procedure that may be considered as belonging to wrapper models. The selection algorithm is very simple; it is based on analysis of connection weights of the matrix **W** that have been formed in the training process at the parameter T value that provides the best recognition rate for the initially generated feature set. The following reasoning serves as foundation for this procedure. The weights of connections directed from the outputs of A-neurons to the inputs of R-neurons (matrix **W**) are modified during the training process (Section 3) by Eqs. (4) and (5). If the final connection weight of some A-neuron is large in absolute value, it means that this A-neuron (and the corresponding LiRA-feature) strongly influences the recognition process and, therefore, is useful. Therefore, we introduce an integral characteristic of usefulness of A-neuron (and the corresponding LiRA-feature) as the sum of absolute weights of connections directed from it to all R-neurons. Let vector **Q** of N components represent this integral characteristic of usefulness of all A-neurons. At the first step, all values $Q_j, j = 1, 2, 3, \ldots, N$ are computed according to the formula

$$Q_j = \sum_{m=1}^{M} |W_{j,m}|, \tag{6}$$

where M is the number of classes (and R-neurons).

At the second step, the most useful A-neurons are chosen in turn. The choice of the first most useful A-neuron (and the respective LiRA-feature) is performed as follows

$$s = \arg \underset{j=1}{\overset{N}{\mathrm{MAX}}} Q_j, \tag{7}$$

where s is the index of the selected A-neuron in the feature pool.

The second most useful A-neuron (and the LiRA-feature) is chosen by Eq. (7) under condition that $j \neq s$, and so on. Selection of the most useful features in descending order is accomplished analogously, until their preset amount is achieved.

Let us underline that application of the described feature selection procedure is reasonable only on condition that the classifier connection structure is optimal, that is, provides high recognition rate. However, formation of optimal connection weights of the matrix **W** at large values of N is rather computationally expensive. Indeed, first it is necessary to find the value of parameter T that provides the best recognition rate. This is done by training the classifier using all applicable values of parameter T in turn and testing the classifier after each convergence of the training process. Second, the connection structure formed at the optimal value of parameter T is analyzed by the above feature selection procedure.

By means of the feature selection procedure, 7000 features were chosen from the previously generated pool of 60 000 features. The selection procedure was performed using the LiRA connection structure formed after the training process convergence at $T = 0.4$. This feature set is named "Selected" and applied in the experiments described below.

As in the previous series of experiments, the LiRA classifier with the 7000 selected features was trained at different values of the T parameter. The classifier error rate was measured for each value of the T parameter after training convergence. The obtained error rates are presented in Table 1 in the row "Selected". Fig. 3 (the upper curve) and Fig. 4 (the lower curve) show the dependence of the LiRA error rate obtained with the 7000 selected features on the parameter T values. The number indicating the total amount of epochs till the classifier convergence is placed near each point of the plot of Fig. 3.

Also, to check performance of the feature selection procedure, 10 sets of 7000 features each were randomly chosen from the same pool of 60 000 initial features. The classifier was trained at different values of the T parameter with each of these 10 random feature set. The classifier error rate was measured for each value of the T parameter after convergence. Results of these experiments are shown in Table 1 in the rows numbered from 1 to 10. The row

Table 1
The MNIST base test set error rate of LiRA using 7000 LiRA-features at different values of the parameter T. Rows 1 to 10 represent the error rates each obtained at certain realization of 7000 random features. The rows "Mean" and "Std" show the averaged error rates of realizations and their standard deviations. The last row "Selected" shows the results obtained using 7000 selected LiRA- features.

No	T									
	0	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45
1	203	198	198	201	188	185	187	188	186	189
2	201	205	194	193	183	176	179	187	187	183
3	204	199	192	188	193	189	190	182	187	190
4	216	200	206	196	197	189	195	197	196	193
5	214	212	194	198	191	194	186	175	179	187
6	205	213	200	208	191	202	200	192	194	197
7	199	201	200	190	187	189	184	187	193	191
8	221	208	210	212	192	204	199	203	201	212
9	212	206	201	194	200	184	177	185	185	191
10	218	198	199	195	205	200	209	195	198	200
Mean	209.3	204.0	199.4	197.5	192.7	191.2	190.6	189.1	190.6	193.3
Std	7.80	5.66	5.52	7.60	6.48	8.83	10.1	7.99	6.85	8.12
Selected	189	190	181	176	169	179	169	173	172	166

"Mean" demonstrates error rates for random feature sets averaged on 10 upper rows.

To assess whether the experimental data represented by "Mean" and "Selected" rows of Table 1 are statistically different from each other Student's t-test was used. Note that the "Mean" row data are averaged on 10 realizations, while the "Selected" row represents data of a single realization. The resulted $t = 13.5$ corresponds to the null hypothesis risk p of only 2.75 E-07, so we conclude that the rows are statistically different.

Fig. 4 graphically demonstrates difference between recognition capabilities of the selected and random feature sets of the same size of 7000 LiRA-features.

The row "Selected" of Table 1, the upper curve of Fig. 3, and the lower curve of Fig. 4 show that at increasing defense space values the LiRA error rate (with the selected 7000 features) decreases and the best result on the test set of the MNIST database equal to 1.66% errors is achieved at $T = 0.45$. Increasing T up to $T = 0.5$ results in non-convergence of the training process both for the selected set of features and for all 10 random sets of 7000 features. The recognition result of 1.66% (166 errors) is somewhat worse than 150 errors committed by LiRA in the previous series of experiments with 60 000 initial features, however both numbers are comparable. At the same time, the feature selection procedure accelerates computational speed of LiRA by about 8 times.

Thus, the experiments presented in this section show that the proposed algorithm provides selection of a rather effective set of features.

6. Modular assembly neural network

Modular neural networks is one of the modern approaches in the field of machine learning (e.g. Muthuraman, Maxwell, & MacLeod, 2003; Nourashrafoddin, Vahdat, & Ebadzadeh, 2007). A modular neural network is understood as a network that consists of analogous and rather independent neural modules.

In a series of papers (Goltsev, 1991, 1996, 2004, 2005; Goltsev & Gritsenko, 2009; Goltsev et al., 2005, 2004; Goltsev & Rachkovskij, 2005; Goltsev & Wunsch, 1998, 2004), modular assembly neural networks have been developed. The modular assembly neural network is intended to classify data belonging to a preset number of classes. Similar modular organization of neural networks is considered in Ezhov and Vvedensky (1996) and Schwenk and Milgram (1994). In the modular assembly neural network, the whole network is divided into several modules (sub-networks) according to the number of classes that the network has to recognize, so that each module represents the corresponding class. Thus, to recognize M classes of objects, the modular assembly neural network should comprise M modules. This division creates favorable conditions for learning, generalization, and adjustment of each class description in its own sub-network (module). Interference with other classes is absent compared to the assembly neural networks where all classes are processed by a single network.

Effectiveness of the modular assembly neural networks was tested on different recognition tasks with a variety of feature types: binary and multi-valued, artificially designed and quite natural (direct use of brightness of image pixels as features) (Goltsev, 2005; Goltsev & Gritsenko, 2009).

A modular assembly neural network that applies LiRA-features is presented below; it is schematically depicted in Fig. 5. Each of M modules of the assembly network consists of N neurons, each neuron represents a single LiRA-feature. The neurons are identically enumerated inside each module, so that the neurons of different modules representing the same feature have the same index.

There is an input layer S in the modular assembly neural network, it is the same as in LiRA. The outputs of the layer S are transmitted to the neurons of each module by means of the same structure of non-modifiable connections denoted by $\mathbf{V}$ in Section 3. All M modules of the assembly network receive the same set of LiRA-features detected in the input image and, therefore, the same initial set of input neural activity is formed in each module. Let us name this set as a pattern of initial neural activity and designate it by a (binary) vector $\mathbf{A}$, as in LiRA.

Learning in the assembly network takes place by modification of connections that link neurons inside the modules. Let us introduce a separate two-dimensional connection matrix $\mathbf{W}^m (N \times N)$ to represent connection weights inside the m-th module. The network connection architecture consists of M such matrices. Initially, all connection weights inside each module are set to zero. In the process of training, the connection weights change considerably and differently in each module.

Fig. 6 illustrates the recognition process in one module of the network. Its lower part depicts initial binary activity of all N neurons of some module (pattern of initial neural activity) represented by the vector $\mathbf{A}$. The upper part depicts multi-valued secondary activity of the same N neurons (represented by vector $\mathbf{E}$) after propagation of the initial neural activity through the trained connection structure $\mathbf{W}$ inside the module. The figure shows that components of the vector $\mathbf{E}$ may take both positive and negative values.

The output layer R of the modular assembly neural network consists of M R-neurons, as in LiRA, so that the neural activity

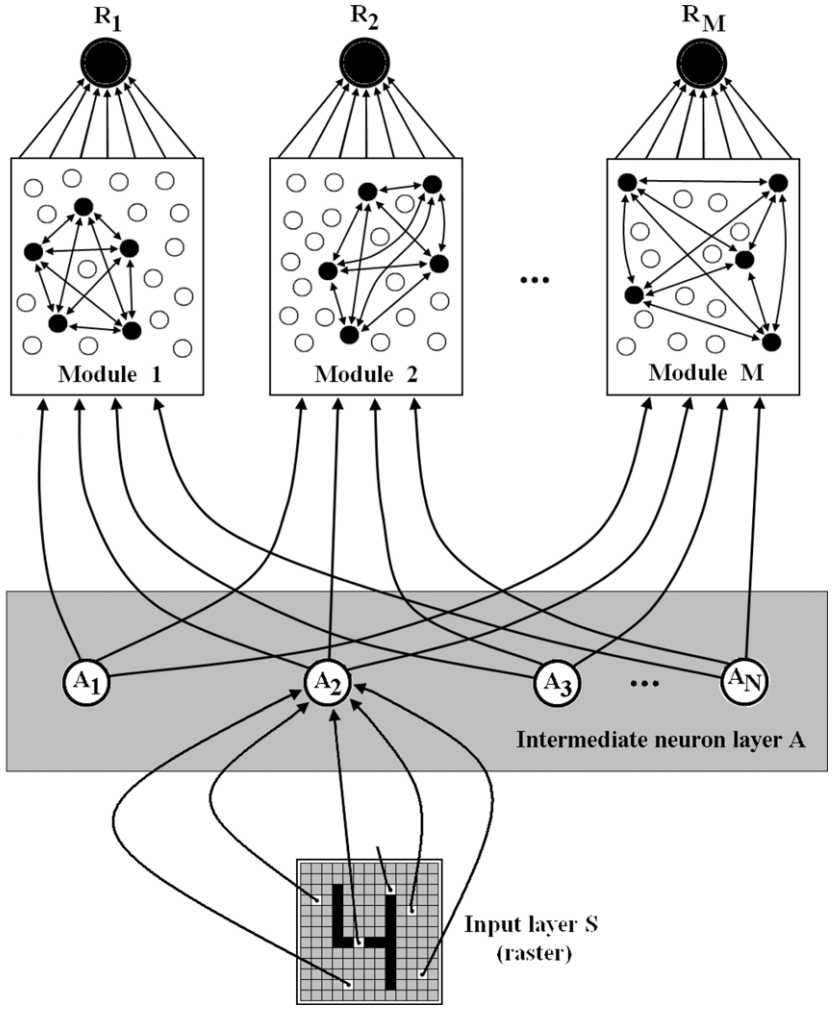

Fig. 5. Schematic picture of a modular assembly neural network. Each pair of mutual connections between two neurons inside the modules is depicted by one line with arrows.

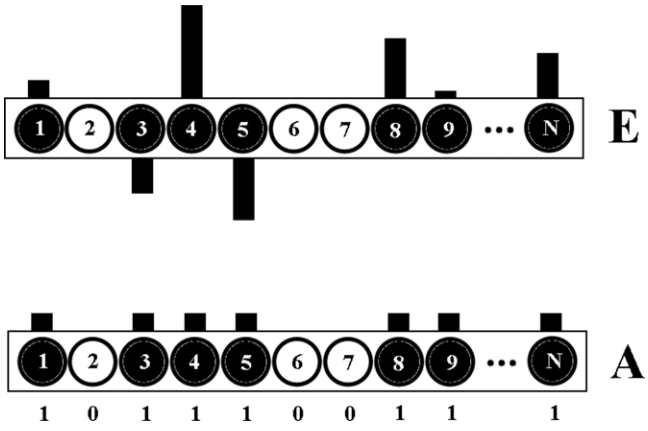

Fig. 6. Two steps of the recognition process in single module of the modular assembly neural network. The lower part (vector **A**) shows a binary pattern of the initial neural activity of all N neurons of some module. The upper part (vector **E**) shows a multi-valued pattern of secondary activity of the same N neurons after propagation of the initial neural activity through the trained connection structure **W** inside the module.

of each module is represented by the corresponding R-neuron. In the recognition procedure, every R_m-th neuron summarizes secondary activities of only those neurons of the m-th module that constituted the pattern of initial neural activity (one-valued components of the binary vector **A**), and represents, ipso facto, their overall activity. So, the activity level of the R_m-th neuron is calculated according to the formula

$$R_m = \sum_{i=1}^{N} A_i E_i^m. \tag{8}$$

R-neuron with the maximal activity defines the recognition result. The index of the class-winner is obtained by Eq. (3).

The training procedure in the assembly network, which is named in Goltsev (2004, 2005), Goltsev and Gritsenko (2009) and Goltsev et al. (2004) as "differentiation" or "secondary learning", is performed as described in Section 3. All available training samples are presented to the network in turn. All LiRA-features are extracted from the current input image by means of the connection structure **V**, and identical patterns of initial neural activity **A** are set in each module of the network. The network tries to recognize the input training sample using the particular value of the defense space T. The training sample belonging to the m-th class is considered as correctly recognized, if the R_m^* value exceeds activity levels of all other R-neurons. In this case, no connection modification is done. However, if it turns out that some wrong R_d-th neuron has maximal activity, then connections in modules m and d are modified according to the following formulas (similar to Eqs. (4) and (5)):

$$W_{j,i}^m = W_{j,i}^m + \Delta W \left(A_j \wedge A_i\right), \tag{9}$$

$$W_{j,i}^d = W_{j,i}^d - \Delta W \left(A_j \wedge A_i\right), \tag{10}$$

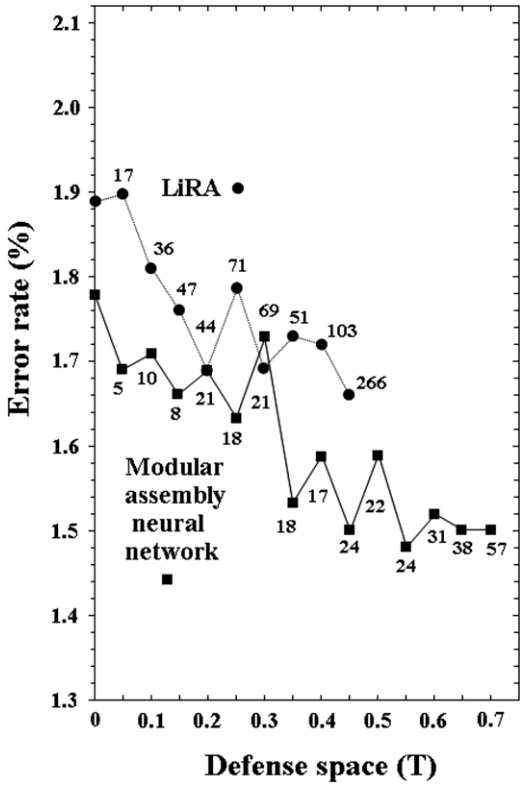

Fig. 7. Dependence of the LiRA classifier error rate (%) and the modular assembly neural network error rate (%) upon the defense space size (parameter T) using the same 7000 selected LiRA-features.

where ΔW is the weight increment value, the same as in formulas (4)–(5), $\wedge$ is conjunction, $i = 1, 2, 3, \ldots, N, j = 1, 2, 3, \ldots, N$.

According to terminology of Hebb (1949), every application of the procedure described by Eq. (9) results in binding together a set of neurons of the m-th module into a neural assembly. During the training process, overlapping neural assemblies are formed inside each module creating some interconnected neural structures which may be considered as a description of the corresponding class.

Evidently, modular assembly neural network demands increasing the number of neurons proportionally to the number of classes that the network has to recognize. On the other hand, this is the price for the network's capability to generalize descriptions of all classes separately within the corresponding modules.

7. Experiments with the modular assembly neural network

The same 7000 LiRA-features selected according to the description of Section 5 and used in the experiments with LiRA were used in the following experiments with the assembly network. The modular assembly neural network was trained at various values of the parameter T using the training set of the MNIST database, as LiRA. After training convergence at the current value of parameter T, the network recognition ability was measured using the test set of the MNIST database. Fig. 7 (the lower curve) demonstrates dependence of the network error rate upon the values of parameter T. The number indicating the total amount of epochs needed for the network convergence for current value of T is placed near each point of the plot. The error rate decreases with increasing T. The best result of the assembly network is 148 errors, it is achieved at $T = 0.55$. When parameter T is increased up to $T = 0.75$, the network training process does not converge.

The best assembly network error rate 1.48% (148 errors) is tangibly less than the best result of 166 errors that LiRA commits in the previous series of experiments using the same 7000 selected LiRA-features. It is also a little bit better than the best error rate of LiRA obtained with the whole initial set of 60 000 LiRA-features (150 errors). In Fig. 7, the assembly network plot is located considerably lower than that of LiRA along the whole its length excluding only two points where the error rates are almost the same. In this case, Student's t-test gives $t = 4.19$ and risk $p = 0.0023$, so we conclude that the modular assembly neural network performs statistically better than the LiRA perceptron.

The numbers of convergence epochs presented in Fig. 7 show that the modular assembly neural network demands considerably less epochs during training versus LiRA, thus having a better ability to form separating surfaces in the large-scale space of LiRA-features.

Therefore, from this experimental comparison we may conclude that classification capability of the modular assembly neural network exceeds that of the LiRA classifier. However, superior classification quality of the assembly network has the cost of considerable decreasing of its computational speed.

8. Discussion and conclusion

The procedure of assembly neural network training described in Section 6 differs from its counterparts presented in Goltsev (2004, 2005), Goltsev and Gritsenko (2009) and Goltsev et al. (2004). In the referred publications, a double-stage mode of the network training is postulated: the first stage is initial learning, and the second stage is differentiation or secondary learning. At the first initial learning stage, each module is trained using all available training samples of the corresponding class according to Eq. (9). The aim is to create in the corresponding module an initial representation of each class from overlapping neural assemblies of the training set samples of this class. In the present paper, this initial learning stage is absent. Preliminary experiments have shown that the recognition rate of both the assembly network and LiRA considerably deteriorates when the initial learning stage is performed. This experimental result suggests that in some cases the initial learning stage is not necessary and may be even rather harmful.

Comments to this fact are as follows. Training of the assembly network (and LiRA) is a process of local minimum finding in the connection weight space at surroundings of some initial point with coordinates that are defined by initial values of the connection weights. In the present work, the initial point is placed at the origin of coordinates (all initial connection weights are equal to zero). Using the initial learning stage shifts this point at some distance away from the origin of coordinates, let us designate this point as L. Thus, the experiments show that for LiRA-features the local minimum value at surroundings of the origin of coordinates is much less than that at surroundings of the point L.

By the way, the modular assembly neural network of the described type can be replaced with an equivalent neural network of perceptron type (although such a replacement does not lead to any improvement). To do so, the following perceptron structure has to be build. The second layer A should consist of N neurons without connections between them, as in LiRA. Every pair of these neurons is represented by one corresponding neuron in the third layer B. Learning connections link every B-neuron with all neurons of the output layer R. This perceptron structure comprises $(N + NM + M)$ neurons (without neurons of S layer) and the same number N^2 of learning connections, as in the modular assemble neural network. In order to confirm equivalence of such a perceptron with the modular assembly neural network, let us note that conjunction in Eqs. (9)–(10) represents the process of detecting pairs of LiRA-features in the input image. The rest parts of these equations are identical to Eqs. (4)–(5) and describe the same

process of connection weight modification. And N^2 connections within one module of the assembly network transmitting the neural activity to a single R-neuron are equivalent to N^2 learning connections of the considered perceptron directed to the same R-neuron.

In this paper, the procedure of feature pool reduction is presented which is based on the analysis of the neural network connection structure formed during the training process. In the handwritten digit recognition task considered in this paper the number of features after feature selection is about ten times less than their initial number, and so the computation speed is increased by about 8 times with retention of comparable recognition capability. The presented feature pool reduction procedure may be used with other types of classifiers and in other tasks. However, the procedure is rather computationally expensive.

In the present work, a direct experimental comparison between the LiRA classifier and the modular assembly neural network is accomplished which shows that recognition capability of the modular assembly neural network exceeds that of classifier LiRA under condition that both classifiers use the same set of LiRA-features. However, let us repeat that the superiority of the assembly network takes place at the cost of its computational speed. It is necessary to note, that direct computer modeling of the modular assembly neural network where each module consists of 60 000 neurons, is impossible on usual PC, because it demands too large RAM. Therefore, using the feature selection procedure is the only way to perform an immediate comparison between both classifiers.

The better recognition capability of the assembly network is such an experimental result that is not evident, but, at the same time, is not unexpected. Indeed, since the assembly network comprises much more learning connections than the LiRA classifier ($MN^2 > MN$), it would be reasonable to expect a better recognition capability. During training process, a multitude of secondary features are formed from LiRA-features in the assembly network that are, in fact, their pairwise combinations. Creation of these secondary features improves generalization process and contributes to formation of more adequate class descriptions in the assembly network modules.

The conclusion about recognition superiority of the assembly network is made on the basis of the experiments where both classifiers use the same realization of 7000 selected LiRA features. This conclusion may be not too much substantiated statistically, but experiments with considerable number of other realizations would require a very long computational time.

Effectiveness of LiRA-features in classification is closely related to their number: the more the number of features, the higher the recognition rate of a classifier. Extraction of a large amount of LiRA-features from an input image in a reasonable time is feasible only owing to absence of any scanning procedures in the extraction process, that is, due to direct using of memorized coordinates of all feature pixels. However, direct using of feature coordinates is possible only for centered images with normalized size of objects, such as digit images of the MNIST database. For other images, extraction of LiRa-features should be somewhat more computationally expensive.

In general, the recognition rate achieved by a classifier depends, first of all, on the feature set used for description of the objects to be recognized. The better the feature set, the higher the performance of the classifier. Moreover, in the field of pattern recognition it is well known that influence of the feature set on the recognition rate is much stronger than that of the classifier type. In the present work, both classifiers apply LiRA-features and experimentally confirm their high effectiveness for the task of handwritten digits recognition of the MNIST database. The obtained results suggest application of these features to solve other tasks where a large number of training samples is available. At the same time, experiments show that the recognition rate strongly depends on such feature parameters as G, H, H^{pos}, H^{neg}. Therefore, a number of questions remain open, including: Why do slight modifications of feature parameters cause so big changes in the classifier recognition capabilities? How to generate the most successful features? Answers to these and other questions may be a subject of a future research.

References

Baidyk, T., Kussul, E., Makeyev, O., Caballero, A., Ruiz, L., Carrera, G., et al. (2004). Flat image recognition in the process of microdevice assembly. *Pattern Recognition Letters*, *25*(1), 107–118.

Baidyk, T., Kussul, E., Makeyev, O., & Vega, A. (2008). Limited receptive area neural classifier based image recognition in micromechanics and agriculture. *International Journal of Applied Mathematics and Informatics*, *2*(3), 96–103.

Baidyk, T., Kussul, E., Makeyev, O., & Velasco, G. (2009). Pattern recognition for micro workpieces manufacturing. *Computación y Sistemas, Instituto Politécnico Nacional, Mexico*, *13*(1), 61–74.

Coates, A., Lee, H., & Ng, A.Y. (2011). An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the 14th international conference on artificial intelligence and statistics, vol. 15*.

Cover, T. M., & Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, *13*(1), 21–27.

Cover, T. M., & Thomas, J. A. (1991). *Elements of information theory*. New York, USA: John Wiley & Sons, Inc.

Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines (and other kernel-based learning methods)*. CUP.

Dong, J. X., Krzyzak, A., & Suen, C. Y. (2003). A fast SVM training algorithm. *International Journal of Pattern Recognition and Artificial Intelligence*, *17*(3), 367–384.

Ezhov, A. A., & Vvedensky, V. L. (1996). Object generation with neural networks (when spurious memories are useful). *Neural Networks*, *9*(9), 1491–1495.

Freund, Y., & Schapire, R. (1999). A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, *14*(5), 771–780.

Goltsev, A. D. (1991). Structured neural network with learning, intended for a textural segmentation of images. *Cybernetics and System's Analysis*, *6*, 149–161 (in Russian).

Goltsev, A. (1996). An assembly neural network for texture segmentation. *Neural Networks*, *9*(4), 643–653.

Goltsev, A. (2004). Secondary learning in the assembly neural network. *Neurocomputing*, *62*(12), 405–426.

Goltsev, A. (2005). *Neural networks with the assembly organization*. Kiev: Naukova Dumka (in Russian).

Goltsev, A., & Gritsenko, V. (2009). Modular neural networks with Hebbian learning rule. *Neurocomputing*, *72*(10–12), 2477–2482.

Goltsev, A., Húsek, D., & Frolov, A. (2005). Assembly neural network with nearest-neighbor recognition algorithm. *Neural Network World*, *15*(1), 9–22.

Goltsev, A., Kussul, E., & Baidyk, T. (2004). A process of differentiation in the assembly neural network. *Lecture Notes in Computer Science*, *3316*, 452–457.

Goltsev, A., & Rachkovskij, D. (2005). Combination of the assembly neural network with a perceptron for recognition of handwritten digits arranged in numeral strings. *Pattern Recognition*, *38*(3), 315–322.

Goltsev, A., & Wunsch, D. C. (1998). Inhibitory connections in the assembly neural network for texture segmentation. *Neural Networks*, *11*(5), 951–962.

Goltsev, A., & Wunsch, D. C. (2004). Generalization of features in the assembly neural networks. *International Journal of Neural Systems*, *14*(1), 39–56.

Hebb, D. O. (1949). *The organization of behavior*. New York, USA: John Wiley & Sons, Inc.

Kasatkina, L. M., Lukovych, V. V., & Pilipenko, V. V. (2006). Speaker recognition with LIRA classifier. *Control Systems and Computers*, *3*, 67–73 (in Russian).

Koller, D., & Sahami, M. (1996). Toward optimal feature selection. In *Proceedings of ICML-96* (pp. 284–292).

Kussul, E., & Baidyk, T. (2002). Improved method of handwritten digit recognition tested on MNIST database. In *Proceedings of the 15-th international conference on vision interface* (pp. 192–197).

Kussul, E., & Baidyk, T. (2003). Permutative coding technique for handwritten digit recognition system. In *Proceedings of international joint conference on neural networks. Vol. 3* (pp. 2163–2168).

Kussul, E., & Baidyk, T. (2006). LIRA neural classifier for handwritten digit recognition and visual controlled microassembly. *Neurocomputing*, *69*(16–18), 2227–2235.

Kussul, E., Baidyk, T., Kasatkina, L., & Lukovich, V. (2001). Rosenblatt perceptrons for handwritten digit recognition. In *Proceedings of international joint conference on neural networks. Vol. 2* (pp. 1516–1520).

Kussul, E., Baidyk, T., & Wunsch, D. (2010). *Neural networks and micro mechanics*. Springer–Verlag.

Kussul, E. M., Kasatkina, L. M., & Lukovich, V. V. (1999). Neural network classifiers for recognition of handwritten symbols. *Control Systems and Machines*, *4*, 77–86 (in Russian).

LeCun, Y. (1998). The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Makeyev, O., Sazonov, E., Baidyk, T., & Martin, A. (2008). Limited receptive area neural classifier for texture recognition of mechanically treated metal surfaces. *Neurocomputing*, *71*(7–9), 1413–1421.

Martin-Gonzalez, A., Baidyk, T., Kussul, E., & Makeyev, O. (2010). Improved neural classifier for microscrew shape recognition. *Optical Memory and Neural Networks (Information Optics)*, *19*(3), 220–226.

Misuno, I., Rachkovskij, D., & Slipchenko, S. (2005). Experimental study of handwritten digits classification. *Systemic Technologies*, *4*(39), 110–133 (in Russian).

Muthuraman, S., Maxwell, G., & MacLeod, C. (2003). The evolution of modular artificial neural networks for legged robot control. *Lecture Notes in Computer Science*, *2714*, 488–495.

Nourashrafoddin, N., Vahdat, A., & Ebadzadeh, M. (2007). Automatic design of modular neural networks using genetic programming. In *Proceedings of the 17th international conference on artificial neural networks* (pp. 788–798).

Rosenblatt, F. (1962). *Principles of neurodynamics. perceptrons and theory of brain mechanisms*. Washington DC: Spartan Books.

Saxe, A. M., Wei Koh, P., Chen, Z., Bhand, M., Suresh, B., & Ng, A. Y. (2011). On random weights and unsupervised feature learning. In *Proceedings of the 28th international conference on machine learning*.

Schwenk, H., & Milgram, M. (1994). Structured diabolo-networks for hand-written character recognition. In: *Proceedings of the international conference on artificial neural networks. Vol. 2* (pp. 985–988).

Vapnik, V. N. (1998). *Statistical learning theory*. New York: John Wiley.

Vega, A., Baidyk, T., Kussul, E., & Pérez Silva, J. L. (2011). FPGA realization of the LIRA neural classifier. *Optical Memory and Neural Networks (Information Optics)*, *20*(3), 168–180.

Yang, Y., & Pedersen, J.O. (1997). A comparative study on feature selection in text categorization. In: *Proceedings of the 14th international conference on machine learning* (pp. 412–420).