Hybrid Neuro-Fuzzy Model with Immune Training for Recognition of Objects in an Image

Mykola Korablyov [1][0000-0002-8931-4350], Natalia Axak [2][0000-0001-8372-8432],
Oleksandr Fomichov [3][0000-0001-9273-9862], Andrii Chuprina [4][0000-0002-4476-8420]

Kharkiv National University of Radio Electronics, Kharkiv, Ukraine
mykola.korablyov@nure. ua, nataliia.axak@nure.ua,
oleksandr.fomichov@nure.ua, andrii.chuprina@nure.ua

Abstract. Modern systems for image processing and analysis are characterized by the active use of artificial neural networks, for training of which, as a rule, gradient methods are used, but their main limitation of the implementation is high computational cost. The use of the principles of hybridization of neural networks, fuzzy logic and evolutionary algorithms allows you to create new types of models that have a higher recognition quality while reducing the computational cost of training. A hybrid neuro-fuzzy recognition model is proposed, which consists of two modules: a convolutional module (CNN) and a neuro-fuzzy classifier module (NFC) built on the basis of a modified ANFIS network. The CNN module, which is trained by the method of back propagation error, acts as a kind of expert system for the NFC module. It is proposed to perform NFC training based on the use of artificial immune systems by presenting all adjustable parameters in the form of a structured adaptive multi-antibody, and consists in adjusting the NFC parameters and structure. Experimental studies have been carried out on test samples, confirming the effectiveness of the proposed model for recognizing objects in an image.

Keywords: object recognition, hybrid model, convolutional module, neuro-fuzzy classifier, immune learning, multi-antibody.

1 Introduction

Today, one of the most rapidly developing scientific and technological areas is image processing and analysis. In recent years, a number of methods, models, and algorithms have been introduced to address these challenges [1], among which artificial neural networks (ANN) are the most effective [2]. There are a large number of standard ANN architectures, and the use of different types of ANN has proven to be quite effective in solving a wide range of problems. The emergence of a new type of ANN - the Convolutional Neural Network (CNN) [3, 4] gives a new impetus in this area of research, it is widely used in various fields, including image recognition, speech recognition, natural language processing, video analysis, etc. [5-12].

When solving applied problems in order to improve accuracy and reduce complexity, the task of finding the optimum neural network (NN) topology and according to it structural (determining the number of hidden layers and neurons in them, interneuronal connections of individual NN) and parametric (adjusting the weight coefficients of NN) optimization. Typically, methods that require the calculation of the gradient of the selected functional are used for NN training [2, 13]. Methods for solving this problem include the organization of parallel and distributed computing on specialized hardware, which can significantly imcrease productivity in solving many practical problems, as well as the use of new evolutionary approaches to the learning of ANN, in particular artificial immune systems (AIS).

The construction of hybrid NN, consisting of different types of NN, each of which is trained by a specific algorithm in layers, in most cases can significantly improve the efficiency of their functioning. The study of the principles of NN hybridization, fuzzy logic and evolutionary algorithms allows us to create new types of models that have a higher quality of recognition in the case of simultaneous reduction of computational costs [14]. One such approach that proposed in this work is hybrid neuro-fuzzy model for recognizing objects in an image on the basis of the CNN and the neuro-fuzzy classifier (NFC), which is taught by AIS.

2 Analysis of approaches to recognize object in the image

The purpose of the recognition procedure is to answer the question: does the object described by the given characteristics relate to the grouping of objects that we are interested in, and if so, to which one. This is a fundamental problem in the computer vision and forms the basis for solving the other tasks, such as the detection, localization and segmentation of objects in the image [15]. Any recognition system includes both the process of synthesizing images, that is, forming descriptions of recognition objects and their classes, and the process of analyzing images, that is, the decision-making process itself. The current level of computing allows to combine in recognition systems both different approaches to describe images as well as methods used in the recognition process.

The use of CNN, especially deep learning models, has influenced the development of methods for detecting and classifying objects in an image [16]. Deep learning models that use several levels of nonlinear information processing to extract and transform features, as well as analyze and classify patterns, have become the leading architecture for most image recognition, classification, and detection tasks [17, 18]. Deep CNN (DCNN), which use GPUs and large datasets for deep learning, have implemented technologies related to various aspects of their improvement [19-22]: 1) network architecture; 2) nonlinear activation functions; 3) mechanisms of regularization; 4) optimization methods, etc. However, the resulting models were often large and slow to calculate.

The main limitations of the known methods and technologies currently in use arise from inefficiency of solving the problem of ANN training, adjusting and adapting to the problem area, processing incomplete and inaccurate source information, interpreting data and accumulating expert knowledge, uniform presentation of incoming information from different sources, etc. Therefore, one of the leading trends is the development of integrated, hybrid systems based on deep learning [6, 14]. Such systems consist of different elements (components), united in order to achieve the set goals. The integration and hybridization of different methods and technologies allows to solve complex problems that cannot be solved on the basis of individual methods or technologies. Today, there is a tendency for hybridization of neural network models and fuzzy logic systems, which combine the ability to represent and process fuzzy knowledge as a base of fuzzy production rules, and the ability to learn on a limited set of examples with a further generalization of the knowledge gained.

This paper aims to develop and study a model that combines ANN, fuzzy logic and AIS technologies for effective image recognition in the image.The result of the research is the creation of a recognition system that, using the developed models as the base architecture, will perform the task of recognizing the objects in an image.

3 Implementation of a hybrid neuro-fuzzy recognition model

A combination of two modules was used to create the hybrid neuro-fuzzy model of recognition: the CNN convolutional module and the ANFIS-based NFC module. The general architecture of the neuro-fuzzy convolution network is presented in Fig. 1. In this hybrid model, the CNN module acts as a kind of expert system for the module with neuro-fuzzy output. CNN receives three channels of 174 x 174 pixel RGB image on its inputs. Thereafter, it goes through four stages of alternating convolution of the input matrix using the 3x3 coagulation kernel and the process of subsampling the maximum values using the 2x2 working matrix. As a result, each of the 4 layers, after completing the full lifecycle, will highlight certain features of the image objects, which as a result of the identifications will be presented as feature maps on the source layer CNN 9x9x128.

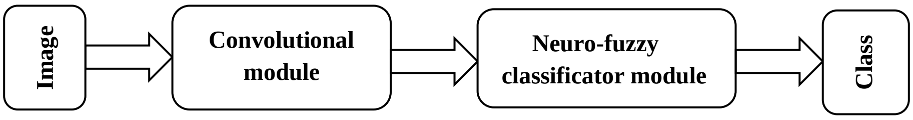

Fig. 1. Hybrid neuro-fuzzy convolution network structure

Each element of the resulting subsample is connected to the input layer of the NFC, where the next step is the phasing process. Each fuzzy signal creates three fuzzy membership functions that characterize the degree of signal saturation (low, middle, high). Based on the activation of production rules from the knowledge base of the NFC, the values of the signals with the highest value are selected, and then the final result of the system operation is formed on the layers of aggregation and dephasing.

3.1 Convolutional module

The CNN convolutional module consists of two types of layers: convolutional layers and subsampling layers. As they alternate, they form the input feature vector for a hybrid neuro-fuzzy classifier. CNN learning algorithm is based on backpropagation.

CNN can run quickly on a sequential machine and learn quickly by parallelizing the convolution process on each card, as well as the reverse convolution when spreading a network error. Compared to other algorithms for processing and synthesizing the features of the objects in the image, CNN uses relatively little preprocessing. For the hybrid pattern recognition model, the architecture of a 4-level convolutional CNN module with one input 3-channel RGB layer was taken (Fig. 1). For convolution feature kernels 3x3 size and subsample maps, the value of which is equal to 2x2 elements were used on all network layers.

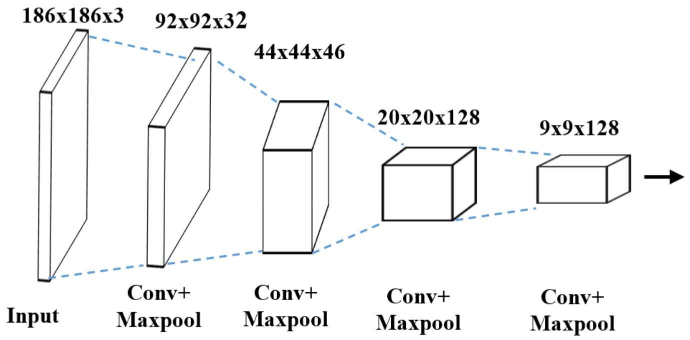

Fig. 2. Structure of convolutional module

The convolutional layer is the main building block of CNN. The layer parameters consists of a set of filters (kernels) for training that have a small receptive field. In the direct pass, each filter performs convolution by rows and columns of the input matrix, calculating the scalar product of the filter and input data, forming a 2-dimensional excitation map of this filter. As a result, the network learns, with the filters activated when the network detects a particular specific type of feature in a particular spatial position at the input. The size of all maps of the convolutional layer is the same and is determined by the formula:

$$(r,c) = f\left(mC - kC + 1, nR - hR + 1\right), \qquad (1)$$

where (r,c) is the calculated size of the convolution card; mC is number of columns the previous map; nR is number of rows of the previous map; kC - number of core columns ; hR – number of core rows.

A kernel is a filter or window that slides across the input map area and finds certain features of objects. The kernel moves through the input map and performs the convolution operation used to process the images:

$$(f*g)[m,n] = \sum_{k,l} f\left(m-k, n-1\right) * g[k,l], \qquad (2)$$

where f – the basic image matrix; g – convolutional kernel. A window of kernel size g passes through the specified step through all the image f, elementwise multiply the contents of the window by kernel g, the result is summed up and written to the original result matrix. In a simplified form, the convolutional layer can be described by the expression:

$$x^{l} = f\left(x^{l-1} * k^{l} + b^{l}\right), \tag{3}$$

where x^{l} – the output of layer l; $f()$ – the activation function; b^{l} – the shift factor of layer l; $*$ – convolution operation of input x with kernel k. In this case, due to the boundary effects, the size of the output matrices decreases:

$$x_{j}^{l} = f\left(\sum_{i}(x_{i}^{l-1} * k_{i}^{l} + b_{i}^{l})\right), \tag{4}$$

where x_{i}^{l} – map of the signs of the *i-th* card (input of layer l); $f()$ – the activation function; b_{i}^{l} – the shift factor of the *i-th* map of layer l; k_{i}^{l} – the convolution kernel of the *i-th* map of layer l; $*$ – convolution operation of input x with kernel k.

In order for the neural network to have good adaptive properties, it must have nonlinear components or perform nonlinear transformations. Given this, within CNN, the weighted sum of inputs (the result of the convolution operation) passes through the ReLU function [6], i.e., each element of the output convolution matrix is an artificial neuron with this type of activation function. Thus, the values in the final object maps are not really the sum of the product of values of the kernels of features and input matrices, but the result of the activation function applied to them.

The subsample layer, like the downsample layer, has maps, but the number is the same as the previous convolutional layer associated with it. The purpose of the layer is to reduce the dimension of the original matrix of the previous layer. If some features have already been detected in the previous convolution operation, then such a detailed image is no longer required for further processing and is compacted to a less detailed one. In the process of scanning with the core of the subsample layer (filter) of the map of the previous layer, the selected areas of the scanning core do not intersect unlike the convolutional layer. Each of them does work related only to their own data area. Each card has a 2x2 size kernel, which allows you to reduce the previous downsample matrices by 2 times. The entire feature map is divided into 2x2 element area, from which the maximum value elements are selected. The work of the subsample layer is described by the formula:

$$x^{l} = f\left(a^{l} * subsample\left(x^{l-1}\right) + b^{l}\right), \tag{5}$$

where x^{l} – the output of the layer l; $f()$ – the activation function; a^{l}, b^{l} – the shift coefficients of layer l; $subsample()$ –operation of selecting local maximum values.

Like all direct signal propagation neural networks, CNN are well trained using the error propagation algorithm [2, 4]. However, the use of this approach in this type of network has its own characteristics. One of the biggest problems with CNN training is the calculation of the δ-error on the subsample layer. This process is represented in several variations within the network. The first case is when the subsample layer is in front of the output signal, then it has neurons and connections of the same type as in the associated layer, so calculating the δ-error is no different from calculating the δ-error of the next layer. The second case is when the subsample layer is in front of the convolutional level. Then the calculation of the δ-error occurs by backward convolution. The purpose of a backward convolution (deconvolution) is to find the equation of a direct convolution given in the form $f * g = h$, where h - is the recorded signal and f is the signal that need to be restored, and it is known that the first signal is obtained by convolution of the second signal with some known signal g. However, if we do not know g in advance, then we need to estimate it. This is usually done by using statistical estimation methods.

Usually, the previous layer after the convolution layer is subsample layer, and according to the task we need to calculate the δ-errors of the current layer (convolutional) on the basis of knowledge about the subsample δ-errors. In fact, the δ-error is not calculated but copied. When the signal is propagated directly, the neurons of the subsample layer are formed by nonoverlapping scan windows over a downsampling layer, in which the neurons with the maximum value are selected. In reverse propagation, the δ-errors are returned only to the neuron that was previously selected with the maximum output signal in its subgroup. Others get a zero δ-error.

3.2 Neuro-fuzzy classifier with immune training

The NFC structure is based on a modified ANFIS network, which is a multilayer network with direct signal propagation that implements the first-order Takagi-Sugeno fuzzy output algorithm for n input variables described by m fuzzy sets [23, 24]:

$$\begin{aligned} &\mathrm{R}_i : IF\ x_i\ is\ A_{i1}\ AND \ldots AND\ x_j\ is\ A_{ij}\ AND \ldots AND\ x_m\ is\ A_{im}\ , \\ &THEN\ y = k_{i1}x_1 + \ldots + k_{ij}x_j + \ldots + k_{im}x_m + k_{i0}, i = 1, \ldots, n\ , \end{aligned} \quad (6)$$

where $x_j, j = \overline{1,m}$ – input variables, $y_i, i = \overline{1,n}$ – output variables A_{ij} – linguistic terms representing fuzzy sets; k_{ij} – coefficients of arguments of function; k_{i0} – shifts.

The network consists of 5 structural layers (Fig. 3). The first layer is the entry level. It consists of *n* nodes, where *n* - the number of inputs to the system. The elements of this layer are combined with the output data obtained from the convolution module. This establishes a link between the two modules – from each element of the feature matrix a separate entry channel into the NFC is allocated. The second layer performs the function of fuzzification the input. The outputs of the neurons of this layer represent the values of the membership functions (MF) $\mu_{A_{ji}}\left(x_j\right)$ for input variables $x_j, j = \overline{1,m}$. A Gaussian MF is used, which is described by the expression:

$$\mu\left(x\right) = exp\left[-\left(\frac{x - c}{\sigma}\right)^2\right], \quad (7)$$

where parameter C denotes the center of the fuzzy set; parameter σ – the coefficient of stretching of the function. The choice of a Gaussian MF is stemmed from its sufficient flexibility and simplicity, which reduces the dimension of the optimization problem when performing fuzzy model training. In this layer are $m * p$ elements, where m – the number of input variables, and p – the number of membership functions. For fuzzification of the input variables, three membership functions are used, which describe the degree of signal saturation, which can vary from 0 to 255.

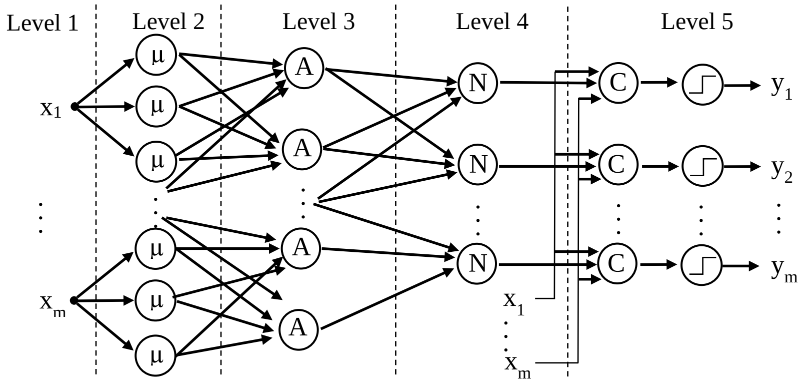

Fig. 3. Structure of neuro-fuzzy classificator

In the third layer, the input signals are multiplied and the weights of the rule (aggregation) are formed:

$$w_i = \prod_{j=1}^{m} \mu_{A_{ij}}\left(x_j\right),\ \ i=\overline{1,p^n}\,, \tag{8}$$

where p^n – number of fuzzy rules.

Each i -th neuron in the fourth layer serves to calculate the ratio of the weight of the *i*-th rule to the sum of the weights of all the rules (normalization):

$$\overline{w_i} = \frac{w_i}{\sum_{i=0}^{m^n} w_i},\ \ i=\overline{1,p^n}\,. \tag{9}$$

The fifth layer implements the function of activation of neurons (consequence), each neuron in it is described by the expression:

$$\overline{w_i} f = \overline{w_i}\left(k_{i0}+k_{i1}x_1+k_{i2}x_2+\ldots+k_{im}x_m\right) = y_i = \begin{cases}1,\ if\ y_i>0,\\ 0,\ if\ y_i\le 0,\end{cases} \tag{10}$$

where $k_{ij}, i=\overline{1,p^n}, j=\overline{1,m}$ – parameters of i -th rule. The outputs of this layer are the outputs from the network, each result of which is the value of the ratio of the result to a particular class, which are predetermined according to the conditions set by the recognition task.

The second and fifth layers are adaptive, the settings of which are made during the training of the network. In the second layer, the MF parameters (centers of fuzzy sets C and stretching factors σ are adjusted for the Gaussian MF), and in the fifth - the parameters of fuzzy rules $k_{ij}, i=\overline{1,p^n}, j=\overline{1,p}$. Some neurons in the NFC are adaptive, and each neuron output depends on the parameters related to that neuron. The learning rule determines how these parameters change to minimize the fuzzy output error.

The NFC training represents the following optimization task:

$$\sqrt{\frac{1}{n*m}\sum_{i=1}^{m}\sum_{j=1}^{n}\left[y_i\left(x_j,P\right)-y_i^r\right]^2}\Rightarrow\min_P, \tag{11}$$

where y_i^r – the required values of the NFC outputs; P – the vector of NFC parameters, c_{ij}, σ_{ij} and $k_{ij}, i=\overline{1,p^n}, j=\overline{1,m}$ are used as parameters, $y_i\left(y_i,P\right)$ – i -th output NFC value with inputs values x_j, dependent from parameter P .

We will perform NFC training on the basis of AIS [25] in order to adjust the parameters of the MF input variables and fuzzy rules coefficients, as well as to synthesize NFC – to remove redundant rules and to corresponding change the network structure. The main idea of teaching NFC with the use of AIS is to present a solvable problem in the form of an antigen, and its possible solutions - in the form of antibodies [26]. We form a population of antigens from the training sample examples $Ag=\left\{Ag_1,Ag_2,\ldots,Ag_M,\right\}$, where M – the size of the antigen population corresponding to the number of examples in the training sample. Each antigen is represented by a fixed length vector: $Ag_i=x_1^i,x_2^i,\ldots,x_n^i,y_1^i,y_2^i,\ldots y_k^i,$, $i=\overline{1,M}$, where $x_1^i,x_2^i,\ldots,x_n^i$ – input variables, and $y_1^i,y_2^i,\ldots y_k^i$ – output NFC variables for i -th example of training sample. In one antibody, all configurable parameters of the NFC are encoded: c_{ij}, σ_{ij} and k_{ij} , $i=\overline{1,p^n}$, $j=\overline{1,m}$.

To train the NFC, a model is used for representing all tunable parameters in the form of a structured adaptive multi-antibody [27, 28], presented in fig. 2 where c_{ij}, σ_{ij} , $i=\overline{1,p^n}$, $j=\overline{1,m}$ – parameters of Gaussian MF for n input variables, each of them has m terms; k_{sv} , $s=\overline{1,n}$, $v=\overline{0,m}$ – coefficients n of fuzzy rules (6).

$c_{11},\ldots,c_{1m},\ldots,c_{n1},\ldots,c_{nm}$; $\sigma_{11},\ldots,\sigma_{1m},\ldots,\sigma_{n1},\ldots,\sigma_{nm}$	$k_{10},\ldots,k_{1n}$	$\cdots$	$k_{v0},\ldots,k_{vn}$
Ab_0	Ab_1	$\cdots$	Ab_{L-1}
Part 1	Part 2		

Fig. 4. Multi-antibody structure

The population of multi-antibodies can be represented as $mAb = \{ mAb_1, mAb_2, \ldots, mAb_n \}$, where $mAb_i = \{ Ab_0, Ab_1, \ldots, Ab_{L-1} \}$, $i = \overline{1, N}$ – i-th multi-antibody, which is a structured vector whose length L changes during the execution of the immune algorithm; N – multi-antibody population size. Each multi-antibody mAb_i , $i = \overline{1, N}$ of the population is characterized by a full set of tunable parameters of NFC. In the structure of the multi-antibody shown in Fig. 4, separation of configurable parameters is used in two independent parts: part 1 with MF parameters c_{ij}, $\sigma_{ij}, i = \overline{1, p^m}, j = \overline{1, m}$, and part 2 with the parameters of fuzzy rules $k_{s,v}, s = \overline{1, n}, v = \overline{0, m}.$

Part 1 of multi-antibody represents a single antibody Ab_0, whose length is fixed, because the number of phase transitions for the input variables is constant. Part 2 of multi-antibody consists of ($L-1$) independent antibodies $Ab_1, \ldots, Ab_{L-1}$, each of them contains coefficients $k_{s,v}, s = \overline{1, n}, v = \overline{0, m}$ for one fuzzy input rule. Part 2 of the multi-antibody is adaptive because in the learning process, in addition to optimizing the coefficients of fuzzy rules contained in this part, their total number also changes, i.e. structural synthesis is performed. Structural synthesis, in turn, leads to a change in the number of neurons in the hidden layers of the NFC (Fig. 3).

A structured method of forming multi-antibodies allows to increase the efficiency of the immune algorithm due to the separate use of immune operators to each part of the antibody - the immune operators are applied to the parameters of the MF and the coefficients of the fuzzy rules separately. The affinity calculation is performed for the multi-antibody as a whole, using both parts. The initial population of multi-antibodies is generated randomly. The size of the population of multi-antibodies is chosen small, because the use of a large number of multi-antibodies in a population leads to large computational costs.

The immune algorithm that implements the training of NFC is based on the principle of clonal selection and the theory of the immune network [25, 26]. The adjustment of the NFC parameters is performed according to the principle of clonal selection. The clonal selection algorithm provides support for the diversity of multi-antibodies in a population by globally viewing the range of tunable parameters, avoiding local minimum. The application of the provisions of the theory of the immune network allows us to evaluate the interaction of antibodies with each other and do a suppression, thus eliminating the redundancy of the fuzzy inference rules. Formally, the immune learning algorithm of NFC can be represented as follows:

$$ImmAlg\left(P^L, L, mAb, N, Ag, M, Op, n_c, N_c, d, \delta_{net}, Alph, A, Aff, gen, t \right) \qquad (12)$$

where P^L – the search range; L – the search range dimension; mAb –multi-antibody population where $mAb = \{ mAb_1, mAb_2, ..., mAb_N \}$; mAb_i – i -th multi-antibody of population mAb: $mAb_i = \{Ab_0, Ab_1, Ab_2, \ldots, Ab_{L-1}\}$; N – multi-antibody population size; Ag – antigen population: $Ag = \{Ag_1, Ag_2, \ldots, Ag_M\}$; M – learning sample size; Op – set of used immune operators where $Op = \{Clone, Mutate, Edit, Suppress\}$; n_{Cl} – number of multi-antibodies for cloning; N_{Cl} – number of clones of one multi-antibody d – the number of multi-antibodies with the worst affinity to be replaced when editing the population of antibodies; δ_{net} – network compression ratio; $Alph$ – the alphabet with which antibodies are encoded; A – the power of alphabet $Alph$; Aff – affinity function; gen – work generation of immune algorithm; t – algorithm termination criterion.

NN training algorithm is an iterative procedure of the sequential identifying observations from the training sample. Antigens are examples of a training sample. Each antibody encodes one possible solution, and the amount of antibodies in the population corresponds to the number of examples in the training set. The training algorithm of NN is the following sequence of steps:

1. Initialization of the initial population of multi-antibodies mAb - performed randomly. Calculation of the affinity of each multi-antibody for antigen

2. Affinity calculation Aff_{mAb-Ag} of each multi-antibody for antigen:

$$Aff_{mAb-Ag} = \left(1 + d_{mAb-Ag}\right)^{-1}, \tag{13}$$

where d_{mAb-Ag} – Hamming distance between the obtained output values of NN $y_i, i = \overline{1,n}$ and desired y_i^r for all S population antigens Ag:

$$d_{mAb-Ag} = \sum\nolimits_{i=0}^{n} y_i, where\, y_i = \begin{cases} 1, \, if \; y_i \neq y_i^r \\ 0, \, if \; y_i = y_i^r \end{cases}. \tag{14}$$

To calculate the affinity of a multi-antibody, it is necessary to substitute the parameters encoded in the multi-antibody into the NFC. The input features x_m are fed to the network input and the values of the output y_s variables are calculated. Thus, the affinity of each multi-antibody mAb_i is calculated in relation to the entire population of antigens Ag.

3. Cloning of multi-antibodies is proportional to their affinity and formation of a clone population Cl. The parameters of the cloning operator are the number of antibodies n_{Cl} clone and the cloning ratio of multi-antibodies N_{Cl}. A fixed parameter value n_{Cl} is used. The cloning ratio of multi-antibody N_{Cl} is regulated during the operation of the immune algorithm, depending on the affinity of the multi-antibody according to the ratio:

$$N_{Cl} = \alpha * N_{Cl\_min} + (1-\alpha) * N_{Cl\_max} \ , \qquad (15)$$

where $\alpha = \dfrac{Aff_{best} - Aff_{mAb-Ag}}{Aff_{best}}$; N_{Cl_min} and N_{Cl_max} – minimum and maximum cloning ratio of multi-antibody; Aff_{best} – the best affinity value in the current generation.

4. Clone mutation is inversely proportional to the affinity of multi-antibodies and the formation a population of mutated clones MC. Mutation of selected multi-antibody parameters mAb performed by adding a gaussian noise:

$$mAb_{i+1} = mAb_i + N(0, \sigma_i) \qquad (16)$$

To change the variance of a random variable σ_i the ratio $\sigma_i = \sigma_i \dfrac{Aff_{best} - Aff_{mAb-Ag}}{Aff_{best} - Aff_{worst}}$ is used, where Aff_{worst} – the worst affinity value in the current generation.

5. Calculation the affinity of a population of mutated clones MC in order to (13). If, as a result of a mutation, affinity improves, then replace with clones the corresponding multi-antibodies in the population mAb.

6. Calculation the affinity of antibodies within part 2 of multi-antibodies. Suppression of antibodies with affinity greater than a predetermined threshold δ_{net}. The affinity calculation is performed in accordance to the expression:

$$Aff_{Ab-Ab} = (1 + |Ab_1 - Ab_2|)^{-1} = \left(1 + \sqrt{\sum_{j=0}^{n} \left(k_{1j} - k_{2j}\right)^2}\right)^{-1} \qquad (17)$$

Perform suppression by removing antibodies Ab_1 with affinity greater than a given threshold δ_{net}, allows to reduce the number of neurons and connections between them in the hidden layer and eliminate their redundancy.

7. Checking the stopping criterion. As a stopping criterion, either the achievement of a given affinity threshold or the achievement of a given number of generations of the algorithm's operation is used. The result of the algorithm will be a multi-antibody with the best affinity by the population, which determines the structure of the NFC and containing its configured parameters.

Parts 1-5 of the algorithm correspond to the principle of clonal selection. At these stages, the algorithm works with both parts of the multi-antibody. Step 6 correspond to the principle of networking. If previously the multi-antibody was processed as a normal antibody, then at this step the work is performed only with part 2 of the multi-antibody, which consists of individual antibodies representing the parameters of fuzzy output rules k_{ij}, $i=\overline{1, p^n}$, $j=\overline{1,m}$. The structure of the NFC (the number of neurons in the hidden layers) is set in accordance to the number of fuzzy rules encoded in the antibody.

The result of the algorithm is an antibody with the best affinity by the population that contains the parameters of the fuzzy output rules and the MF of the input variables.

4 Experimental results

Ready-made architectural solutions (libraries) were used to build the models for each individual module. The generation of the CNN module was performed based on keras-sharp library which is used to generate deep learning networks. In its turn, the NFC module was based on the fuzzy-class-net software package. But its training was done using AIS. System that was obtained as a result of compilation of two modules was run on a computer that had the following characteristics:

– OS Windows 10;
– the number of physical cores – 2;
– RAM – 4 GB;
– CPU – Intel core i5-3210M 2500МГц.

For the objective analysis of the results, the obtained data of the investigated hybrid neuro-fuzzy model were compared with similar output values obtained using the standard CNN network. This approach makes it possible to evaluate the advantages and disadvantages of the proposed model.

A set of 174 x 174 pixel images was selected for the training. Each image has one target object that needed to be identified during the test experiments. The entire sample of images is divided into two classes: class 1 - cats images and class 2 dogs images. Each set of test data images consists of 1000 instances. 750 copies of each set of test data were randomly selected to train the networks. And for testing the model - 250 copies remaining in each of the sets, plus 250 were randomly selected from those that are already used for training in both classes.

The results of the training are shown in Fig. 5, where the error values are on the *Y* scale and a number of epochs on the *X* scale. As we can see from Fig. 5, during the course of 1500 training epochs, the total error value produced by the hybrid NFN stopped at 0.2. For more accurate metrics, you need to increase the number of initial training sample and the number of training epochs. From the 350 epoch relatively correct answers began to form at the output of the classifier with an accuracy of the result in the range of 65 - 80%. In turn, for standard CNN, more accurate output values began to form from the 300 epoch, and their values ranged from 64 to 79%. In the last 1500 epoch, the value of network error was approximately the same.

As a result of network training, one feature was noticed. The time to perform each training iteration of the hybrid NFN epoch was approximately 5 percent greater than time of a standard CNN network. This is due to the fact that during the training of the hybrid NFN, at the time of neuro-fuzzy classifier training, additional operations are performed to adjust the structure and parameters of the production rules.

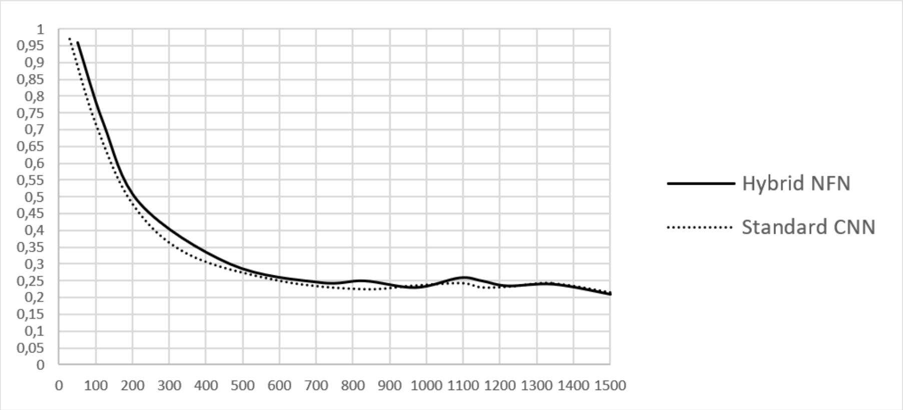

Fig. 5. Graph of training of hybrid NFN and standard CNN

The next phase of the experiments was to test the trained systems on how they would recognize the test data set, which consisted of new, not involved in training and used images. Each model input received 500 images from each class. The following results were obtained as a result of the test. The accuracy of object recognition on the image for the hybrid NFN was 80%, and in the CNN network the ratio was fixed at 78%. Detailed information on this parameter for each of the test classes is given in Table. 1.

Table 1 – Comparison of the recognition accuracy of classes

Image class	Class 1		Class 2	
Model of recognition	Standard CNN	Hybrid NFN	Standard CNN	Hybrid NFN
Recognition accuracy (in %)	77.27%	80.12%	78.49%	79. 84%
Number of recognized images	372	398	374	388

The other equally important indicator is the number of images on which object classes were correctly recognized. In order for an object to be recognized, the system must give an accuracy value greater than or equal to 65% in this test. As a result of the experiment, it was found that for the hybrid NFN, it was 786 precisely found objects, and for CNN this figure was 746 objects. The final indicator difference is 40 images.

From the results of the experiments, it follows that the hybrid NFN algorithm developed slightly slower than the CNN algorithm during the training. Because when you perform the training iteration, additional calculations are performed to set up production rules, which takes up to 5% of the time. But the proposed model did better in recognizing the objects in the image. It was found that the result of image recognition accuracy is greater than that of CNN by 1.5% and the number of correctly recognized images is greater by 40 elements, which is 8% of the total sample of test data.

4 Conclusion

The construction of hybrid NN, consisting of NN of different types, each of which is trained according to a specific algorithm, makes it possible to increase the accuracy and reduce the complexity of solving practical problems, including the problem of recognizing objects in an image. The use of NN hybridization principles, fuzzy logic and evolutionary computations allows the creation of new types of models that have a higher quality of recognition while reducing computational training costs.

To solve the task of recognizing objects in an image, it is proposed to use a hybrid neuro-fuzzy model, which consists of two modules: a convolutional module and a neuro-fuzzy classifier (NFC) module. The convolutional module is implemented on the basis of CNN, which is trained using the algorithm of back propagation error. The NFC module is implemented on the basis of a modified ANFIS network.

It is proposed to train the neuro-fuzzy classifier on the basis of the immune approach by adjusting its parameters and structure. A model which is proposed for the representation of all tunable network parameters in the form of a structured adaptive multi-antibody, consisting of two parts. The structured method for the formation of a multi-antibody allows to increase the efficiency of immune algorithms due to the separate application of immune operators to each part of the multi-antibody and thus solve the task of synthesizing a neuro-fuzzy classifier.

Experimental studies have shown that the proposed hybrid neuro-fuzzy model showed better performance in object recognition in an image than the standard CNN.

References

1. Duda R.O. Pattern classification / R.O. Duda, P.E. Hart, D.G. Stork. – Wiley & Sons, 2010. – 738 p.
2. Khaikin S. Neural networks. Full course: A Comprehensive Foundation, 2nd edn, Williams, Moscow, 2006. – 1104 p.
3. LeCun, Y. Convolutional networks for images, speech, and timeseries I Y. LeCun, Y. Bengio II The Handbook of Brain Theory and Neural Networks. 1995. pp. 255-258.
4. GoodfellowI. Deep learning / I. Goodfellow, Y. Bengio, A. Courville: –The MIT Press, 2016. – 800 p.
5. Zheng YI. Evaluation and Implementation of Convolutional Neural Networks in Image Recognition / First International Conference on Advanced Algorithms and Control Engineering. IOP Conf. Series: Journal of Physics: Conf. Series 1087 (2018).
6. Waseem Rawat, Zenghui Wang. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review / Neural Computation 29, 2017, pp. 2352–2449.
7. Qingdong Wei, Fengjing Shao, Ji Lin. Research Summary of Convolution Neural Network in Image Recognition / Proceedings of the International Conference on Data Processing and Applications, 2018, pp. 39–44.
8. Jason Brownlee. Convolutional Neural Network Model Innovations for Image Classification / Deep Learning for Computer Vision, 2019.
9. Venkatesan R., Li Baoxin. Convolutional Neural Networks in Visual Computing, CRC Press, 2017.
10. Keunyoung Park, Doo-Hyun Kim. Accelerating Image Classification using Feature Map Similarity in Convolutional Neural Networks / Journals Applied Sciences, Vol. 9, Iss. 1,

2018.

11. Jungmo Ahn, JaeYeon Park, Donghwan Park, Jeongyeup Paek, JeongGil Ko. Convolutional neural network-based classification system design with compressed wireless sensor network images / Published: May 8, 2018.
12. Mingyuan Xin, Yong Wang. Research on image classification model based on deep convolution neural network /EURASIP Journal on Image and Video Processing, Article number: 40, 2019.
13. Y. Lecun ; L. Bottou ; Y. Bengio ; P. Haffner Gradient-based learning applied to document recognition / Proceedings of the IEEE (Volume: 86 , Issue: 11 , Nov. 1998), **.** 2278 – 2324.
14. Castillo O. Melin P. Hybrid Intelligent Systems in Control, Pattern Recognition and Medicine / Studies in Computational Intelligence. Springer Nature Switzerland AG, 2020, 381 p.
15. Karpathy, A., & Fei-Fei, L. Deep visual-semantic alignments for generating image descriptions / In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Red Hook, NY: Curran. 2016. pp. 3128–3137.
16. Boukaye BoubacarTraore[ab]BernardKamsu-Foguem[a]FanaTangara Deep convolution neural network for image recognition / Ecological Informatics // Vol. 48, 2018, pp. 257-268.
17. LeCun, Y. Handwritten digit recognition with a backpropagation neural network IY. LeCun, B. Boser, J. S. Denker, D. Henderson, R. Howard, W. Hubbard, L. Jackel //Advances in Neural Information Processing Systems. 1990. № 2. pp. 396-404.
18. LeCun Y., Bengio Y., Hinton G, Deep Learning / in Nature 521 (7553): 436-44, 2015.
19. Krizhevsky, A., Sutskever, I., & Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks / Part of: Advances in Neural Information Processing Systems 25 (NIPS 2012), Red Hook, NY, 2012, pp. 1097–1105.
20. Li Deng, Dong Yu. Deep Learning: Methods and Applications / MSR-TR-2014-21, 2014.
21. Karen Simonyan, Andrew Zisserman. Very Deep Convolutional Networks for Large-scale Image Recognition / arXiv: 1409.1556, 2014.
22. Matthew D. Zeiler, Rob Fergus. Visualizing and Understanding Convolutional Networks / Computer Vision – ECCV 2014, pp 818-833.
23. Zadeh, Lotfi A. A Theory of Approximate Reasoning. In: Hayes, Jean E. Michie, Donald and L. I. Mikulich (eds.): Machine Intelligence, vol. 9 (based on the International Machine Intelligence Workshop), New York: Elsevier, 1979, pp. 149–194.
24. Kasabov N. The Evolution of the Evolving Neuro-Fuzzy Systems: From Expert Systems to Spiking-, Neurogenetic-, and Quantum Inspired / A Bradford Book The MIT Press Cambridge, Massachusetts London, England, 2013, pp. 165-175.
25. Dasgupta D. Immunological computation, Theory and applications / D. Dasgupta, L.F. Nino – Taylor & Francis Group, 2009. – 278 p.
26. Dasgupta D. Recent Advanced in Artifical Immune Systems: Models and Applications / D. Dasgupta, S. Yu, F. Nino // Applied Soft Computing. Elsevier, 2011. – P. 1574-1587.
27. Korablev N. Immune Approach for Neuro-Fuzzy Systems Learning Using Multiantibody Model / N. Korablev, I. Sorokina // ICARIS 2011, Springer Lecture Notes in Computer Science. – 2011. – Vol. 6825. – P. 395–405.
28. Korablyov M., Axak N., Soloviov D. Hybrid evolutionary decision-making model based on neural network and immune approaches // 2018 IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT) 2018. 2018. V.1. P. 378–381.