Neural Networks in Intelligent Analysis Medical Data for Decision Support

Vasyl Sheketa[a], Mykola Pasieka[a], Nelly Lysenko[b], Oleksandra Lysenko[b], Nadia Pasieka[b] and Yulia Romanyshyn[a]

[a] *National Tech. University of Oil & Gas, Ivano-Frankivsk, 76068, Ukraine*
[b] *Vasyl Stefanyk Precarpathian National University, Ivano-Frankivsk, 76000, Ukraine,*

Abstract
The main purpose of the work was to consider the problem of neural networks and their application, especially for data management and control in the medical industry. The software product, analyzes processing of unstructured and poorly structured medical data reliability, to support decision-making, implements the neural network, was developed and studied from sets of user-defined information flows. On the basis of the scientific task, the program training algorithm was developed, which provides comprehensive support for decision-making based on the study. The developed software application is focused on cross-platform, and the graphical interface is implemented using Java FX. The software product provides a network for the reverse propagation of neural network errors (BackPropagation) and a network of directed random search (Directed Random Search). Designed neural network is trained and further recognizes the type of distribution (uniform, normal) on the specified characteristics, and used the rule "3 Sigma" to generate synthetic data. According to the study, we can conclude that the Directed Random Search learning algorithm, although more complex to implement the search for relevant medical documents, works much faster than the classical reverse distribution.

Keywords 1
Neural network, mathematical models, systems architecture, software applications, CEUR-WS

Introduction

1.1. Basic concepts of neural networks

Artificial neural networks – mathematical models, as well as their software and hardware implementation, built on the principle of biological neural networks - networks of nerve cells of a living organism. Systems, architecture and principle of operation are based on analogy with the brain of living beings. The key element of these systems is an artificial neuron as an imitation model of the brain nerve cell, a biological neuron. This term arose when studying the processes occurring in the brain and attempting to simulate these processes. The first such attempt was the McCalock and Pitts neural networks [1, 4, 6, 11, 12, 18, 23, 27, 34, 37]. As a consequence, after the development of training algorithms, the obtained models were used for practical purposes: in forecasting tasks, for pattern recognition, in control tasks, and others [21]. The neural networks can be classified by:

Type of input information:

- analog neural networks (use information in the form of real numbers);
- binary neural networks (operate with information presented in binary form).

Character of learning:

IDDM'2020: 3rd International Conference on Informatics & Data-Driven Medicine, November 19–21, 2020, Växjö, Sweden
MAIL: vasylsheketa@gmail.com (A. 1); pms.mykola@gmail.com (A. 2); lysenkowa@gmail.com (A. 3); leuro@list.ru (A. 4); pasyekanm@gmail.com (A. 5); yulromanyshyn@gmail.com (A. 6)
ORCID: 0000-0002-1318-4895 (A. 1); 0000-0002-3058-6650 (A. 2); 0000-0002-1029-7843 (A. 3); 0000-0002-1029-7843 (A. 4); 0000-0002-4824-2370 (A. 5); 0000-0001-7231-8040 (A. 6)

- learning with a teacher - known neural network output;
- teacher less learning - the neural network processes only the input of unstructured and poorly structured medical data and generates the output results itself. Such networks are called self-organizing;
- teacher-supported learning - a system of fines and incentives from the environment.

The nature of synapses setting:

- fixed-linked networks (neural network weights are selected immediately based on the conditions of the task, with: $\frac{\partial W}{\partial t} = 0$ where W is the network weights);
- networks with dynamic links (for them, synaptic links are set up in the course of training, i.e., $\frac{\partial W}{\partial t} \neq 0$ where *W* are the weights of the network).

1.2. Back propagation network

Backpropagation is a method of teaching multilayer perceptron. The method was first described in 1974 by A.I. Galushkino and independently and simultaneously by Paul G. Verbos. It was further developed in 1986 by David I. Rumelhart, J.E. Hinton and Ronald J. Williams and independently and simultaneously by S.I. Bartsevim and V.A. Okhoninim (Krasnoyarsk Group). This is an iterative gradient algorithm that is used to minimize the error of multilayer perceptron operation and to obtain the desired output. The main idea of this method is to distribute error signals from the network outputs to its inputs, in the direction opposite to the direct distribution of signals in normal operation. Barth and Ohanian proposed a general method («duality principle»), applicable to a wider class of systems, including systems with a delay, distributed systems, etc. [17] To be able to apply the method of inverse error propagation, the neuron transfer function should be differentiated [24]. The method is a modification of the classical gradient descent method. Error reverse propagation algorithm is one of the methods to teach multilayer forward propagation neural networks (multilayer perceptron's) [2, 7, 25, 30, 36 40]. Training in the method of error propagation reverse involves two passes through all layers of the network: forward and reverse. In a forward pass, the input vector is fed to the input layer of the neural network and then propagated through the network from layer to layer. As a result, a set of output signals is generated, which is the actual reaction of the network to this input image. All synaptic weights of the network are fixed during the direct pass. During the reverse pass, all synaptic weights are adjusted according to the error correction rule, namely: the actual network output is subtracted from the desired one, resulting in an error signal. This signal is then propagated through the network in the opposite direction to the synaptic links. Hence, the name is the method of error propagation backwards. Synaptic weights are adjusted to bring the network output as close to the desired one as possible [9, 10, 13]. The appearance of the reverse propagation algorithm has become a landmark event in the field of neural network development, as it implements a computationally efficient method of multilayer perceptron training. It would be wrong to say that the error reverse propagation algorithm offers a really optimal solution to all potential problems, but it has dispelled the pessimism about multilayer machine learning [26, 29, 33]. Let us consider the work of the algorithm in more detail. Let's assume that it is necessary to teach the next neural network (Fig. 1) by applying the algorithm of error back propagation:

The following symbols are used in this picture:

- each layer of the neural network has its own letter, e.g. the input layer has its own letter a and the source layer c;
- all neurons of each layer are numbered in Arabic numerals;
- w (*a1-b1)* - synaptic scales between neurons *a1* and *b1.*

As an activation function in multilayer perceptron's, the sigmoid activation function, in particular the logistic one, is usually used:

$$OUT = \frac{1}{1 + \exp(-aY)} \tag{1}$$

where a is the tilt parameter of the sigmoid function.

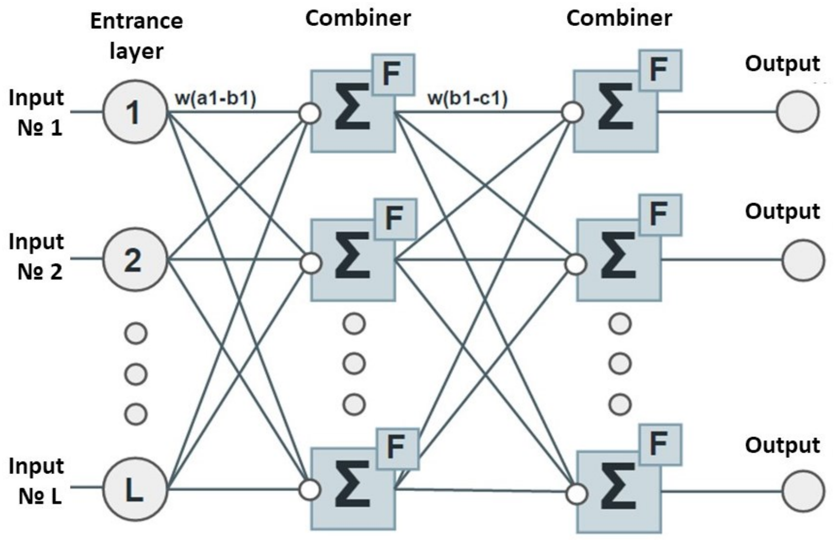


Figure 1: Example of a two-layer perceptron

By changing this parameter, you can build functions with different steepness. Let us agree that for all subsequent considerations will be used exactly the logistic activation function (Fig. 2), represented by the (formula 1).

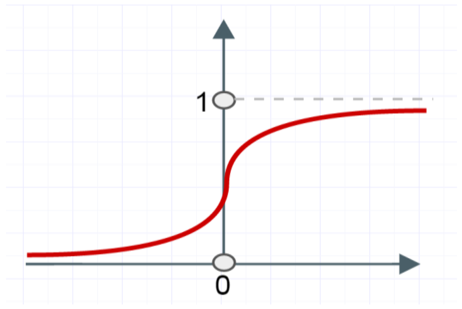


Figure 2: Sigmoid

The sigmoid narrows the range of change so that the OUT value lies between zero and one. Multilayer neural networks have more reflective power than single-layer networks only when non-linearity is present. The compression function provides the necessary non-linearity. In fact, there are many functions that could be used. For an algorithm to reverse the error propagation, it is only necessary that the function is differentiated everywhere. The sigmoid meets this requirement. Its additional advantage is the automatic gain control. For weak signals (i.e. when OUT is close to zero) the input/output curve has a strong slope which gives a high gain. When the signal becomes larger the gain drops. Thus large signals are perceived by the network without saturation, while weak signals pass through the network without excessive attenuation. The purpose of error propagation training for the network is to adjust the weights so that a certain number of inputs lead to the required number of outputs. For short, these sets of inputs and outputs are called vectors. At training it is supposed that for each input vector there is a target vector paired to it, which sets the required output. Together, they are called a training pair. The network learns on many pairs.

The algorithm of error back propagation is as follows:

1. Initialize synaptic weights with small random values.
2. Select the next training pair from the training set; submit the input vector to the network input.
3. Calculate the network output.
4. Calculate the difference between a network output and the required output (target vector of a training pair).
5. Adjust the network weights to minimize the error.

6. Repeat steps 2 to 5 for each vector of a training set until the error on all set reaches an admissible level.

The operations performed in Steps 2 and 3 are similar to those performed at operation of the already trained network, i.e. input vector is supplied and output is calculated. Calculations shall be performed layer-by-layer. In Fig. 1, the outputs of neurons of layer *B* (layer *A* of the input layer, which means there are no calculations in it) are calculated first, then they are used as inputs of layer *C*, the outputs of OUT (C_N) of neurons of layer *C* are calculated, which form the output vector of the OUT network. Steps 2 and 3 form the so-called «pass forward» because the signal is distributed in the network from input to output.

Steps 4 and 5 make up a «reverse pass», here the calculated error signal spreads back through the network and is used to adjust the weights.

Let us consider in detail step 5 - correction of the network scales. Two cases should be highlighted here.

Case 1. Correction of synaptic weights of the output layer

For example, for the neural network model in Fig. 1., these scales have the following designations: $w(B_1\text{-}C_1)$ and $w(B_2\text{-}C_1)$. Let us define that the index *p* will denote the neuron from which the synaptic scales follow, and the *q*-neuron into which it enters [8, 19].

Enter the value Δ, which is equal to the difference between the required T_q and the real OUT_q outputs multiplied by the derivative of the activation logistic function (formula 1.):

$$\Delta = \mathrm{OUT}_q\left(1\text{-}\,\mathrm{OUT}_q\right)\left(\mathrm{T}_q\text{-}\,\mathrm{OUT}_q\right) \tag{2}$$

Then, the weights of the output layer after the correction will be equal:

$$W_{p-q}(\mathrm{i}+1) \;=\; W_{p-q}(\mathrm{i}) + \mathrm{n}\Delta_q\mathrm{OUT}_p \tag{3}$$

where *i* is the number of the current learning iteration; W_{p-q} is the value of synaptic weight connecting neuron *p* with neuron *q*; *n* - the «learning rate» coefficient, which allows to control the average value of weight change; OUT_p - neuron *p* output.

Here is an example of calculations for synaptic weight W_{b1-c1}:

$$\Delta_{c1} = \mathrm{OUT}_{c1}(1\text{-}\,\mathrm{OUT}_{c1})(\mathrm{T}\text{-}\,\mathrm{OUT}_{c1}) \tag{4}$$

$$W_{b1-c1}(\mathrm{i}+1) = W_{b1-c1}(\mathrm{i}) + \;\mathrm{n}\Delta_{c1}\mathrm{OUT}_{b1} \tag{5}$$

Case 2. Correction of synaptic weights of the hidden layer.

For the neural network model in Fig. 1, it will be the corresponding weights between layers *A* and *B*. Let us define that the index p will indicate the neuron from which the synaptic weight follows, and *q* - the neuron which enters (we pay attention to the appearance of a new variable *k*):

$$\Delta_q = \mathrm{OUT}_q\left(1\text{-}\,\mathrm{OUT}_q\right)\sum_{k-1}^{m}\Delta\mathrm{kwq} - \mathrm{k} \tag{6}$$

Then, the weights of the hidden layer after the correction will be equal:

$$W_{p-q}(\mathrm{i}+1) = W_{p-q}(\mathrm{i}) + \;\mathrm{n}\Delta_p\mathrm{OUT}_p \tag{7}$$

Here is an example of a calculation for synaptic weight W_{A1-B1}:

$$\Delta_{B1} = \mathrm{OUT}_{B1}(1\text{-}\,\mathrm{OUT}_{B1}) \tag{8}$$

$$W_{A1-B1}(\mathrm{i}+1) = W_{A1-B1}(\mathrm{i}) + \;\mathrm{n}\Delta_{B1}\mathrm{OUT}_{A1} \tag{9}$$

For each neuron in the hidden layer, Δ must be calculated and all weights associated with this layer must be set. This process is repeated layer by layer until all weights have been corrected.

Despite the many successful applications of inverse propagation, it is not a universal solution. What causes most trouble is an indefinitely long learning process. In complex tasks, it may take days or even weeks to train a network and it may not learn at all. The reason may be one of the following.

Network fixation. In the process of learning a network, the weights may become very large values as a result of correction. This can cause all or most neurons to function at very high *OUT* values, in an area where the compression function derivative is very small. Since the error sent back in the learning process is proportional to this derivative, the learning process can almost freeze. In theory, this problem is poorly understood. Usually this is avoided by reducing the step size of η, but this increases the learning time. Different heuristics have been used to prevent loops or to recover from them, but so far they can only be seen as experimental.

Local minimums. Reverse propagation uses a type of gradient descent, i.e. the descent down the error surface, continuously adjusting the weights in the direction of the minimum. The error surface of a complex network is severely cut and consists of elevations, valleys, folds in the space of high dimensionality. A network can hit the local minimum (shallow valleys) when there is a much deeper minimum nearby. At the point of the local minimum, all directions lead upwards and the network is unable to get out of it. The main difficulty in training neural networks are precisely the methods of getting out of the local minimum: each time, based on the local minimum is again looking for the next local minimum by the same method of reverse propagation of the error until it is no longer possible to find an exit. Step size. A careful analysis of the evidence of convergence shows that the correction of weights is assumed to be infinitely small. It is clear that this is not feasible in practice, as it leads to infinite learning time. The step size should be taken as a final one. If the step size is fixed and very small, the convergence is too slow; if it is fixed and too large, you may experience paralysis or constant instability. Effectively increase the step until the improvement of the evaluation in this direction of the anti-gradient stops and decrease if such improvement does not occur. P. D. Wasserman described an adaptive step selection algorithm that automatically corrects step size during training. The book by A. N. Gorban offers a branched technology of learning optimization.

It should also be noted that the possibility of network retraining is rather a result of erroneous design of its topology [5]. If there are too many neurons the property of the network to generalize information is lost. The whole set of images provided for learning will be studied by the network, but any other images, even very similar ones, can be classified incorrectly. [3, 14, 15]]

1.2.1. Extracting knowledge from a dataset to determine the distribution type (Normal, uniform)

The results of any measurement or observation, presented as figures, can be considered random values corresponding to probable laws. They are likely to understand that it is fundamentally impossible to obtain the actual value of the parameter we are interested in — too many factors affect it, the process of change, etc. We can only get closer to the actual value, estimate the interval in which it falls. All conclusions made when working with random variables are not definitions but probabilistic. An arbitrary random value is most fully described by the distribution function, which determines the probability that a given value will take a value less than or equal to a given one as a result of a single experiment. If a random variable is continuous, its derivative will be a probability density function, and it is impossible to calculate it explicitly for each natural object because of the large set. It is known from the experience of long-term use of the apparatus of applied mathematical statistics that the absolute majority of random phenomena in nature describe functions of only a few types with high accuracy [31]. They are well known; I am called in detail the basic laws of distribution of random variables. Among them, there is one — the normal law of distribution. It is one of the most common models, and the specific features of the function that describes it make this law the main one in applied statistics methods. The normal distribution density graph is characterized by symmetry. This means that deviations from the most probable value of a random value are equal to both greater and lesser ones, a property that simplifies calculations. Most of the methods described prove that the random value under study is distributed by a normal law. Therefore, at the beginning of any statistical analysis processing of unstructured and poorly structured medical of data at least approximately determine the distribution law and estimate the degree of its deviation from the normal [20, 38, 39]. There are many methods that solve the medical problem. The simplest method is based on the visual evaluation of the distribution of asymmetry and excess coefficients by histograms and value. The histogram is a simplified model of the curve of random value density distribution. By constructing it and comparing it with reference plots of the basic laws, we can roughly judge the degree of similarity between them. To build a histogram, a random value is broken down into a certain number of bits (grouping intervals) and counts how many of them fall into each bit. Then, the abscissa is placed on the abscissa axis and the frequencies corresponding to them are placed on the ordinate axis. There are no absolutely strict methods of determining the number of discharges. Mostly 8-12 bits are used. There is no ideal match for histograms of real random variables. By the type of the built histogram we can judge about the degree of deviation from its normal distribution. If the histogram is symmetric with respect to the vertical axis passing through the apex, we can speak about possible approximation of its normal laws. The discrete random value is the most probable value, while continuity is the value where the density of distribution is maximum. If a curve of the distribution law has more than one maximum, the distribution is called binary or poly-modal, respectively. A media random value is a value in relation to which a random value is equally likely to be detected more or less than that value.

1.2.2. Rule "3 sigma"

A normal distribution, also called a Gaussian distribution, is a distribution of probabilities defined by the probability density function, which coincides with the Gauss function:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{(x-\mu)^2}{2\sigma^2}} \qquad (10)$$

where μ - mathematical expectation, σ^2 – the variance of a random variable.

The central limit theorem states that the normal distribution occurs when a given random variable is the sum of a large number of independent random variables, each of which plays an insignificant role in the formation of the whole sum. For example, the distance from a projectile hitting a target with a large number of shots is characterized by the normal distribution. The standard normal distribution is called normal distribution with mathematical expectation $\mu = 0$ and standard deviation $\sigma = 1$.

The rule of 3 sigma (3σ) - where almost all values of a normally widespread random value lie in the interval $[x - 3\sigma; x + 3\sigma]$ (Fig. 3).

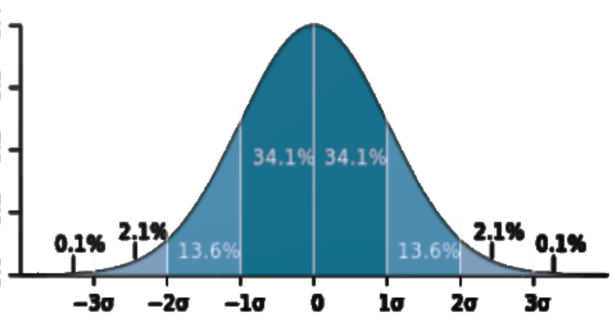


Figure 3: Rule of 3 sigma

To be more precise - at least with 99.7% reliability, the value of a normally distributed random value lies within the specified interval (unless the value of x is precisely known and not obtained as a result of sample processing). If the true value of the value is unknown, then you should use not σ, but s. Thus, the rule of 3 s will turn into the rule of 3 s.

2. Practical Implementation of the Method of Searching for Weakly Structured Information

2.1. UML class diagrams

The GUI package (classes that implement the interface) includes 2 classes: MainController and Main. MainController is a class responsible for user interaction; it contains event handlers and the main interface elements (buttons, tables, combo boxes), as well as links to Main class. The Main class is responsible for initializing the main window of the program and loading the interface structure from an FXML file [16, 32].

The neuralNets package contains the NeuralNet interface and the DirectedRandomSearchNet and BackPropagationNeuralNet classes (Fig. 5). The NueralNet interface is used (according to SOLID principles) to provide flexibility of the program architecture. It contains signatures of the following methods: train - training of the neural network, solve - calculation of results by already trained neural network, loadWeights - loading of scales from a file, saveWeights - saving of scales to a file, getWeights - return of scales, and others.

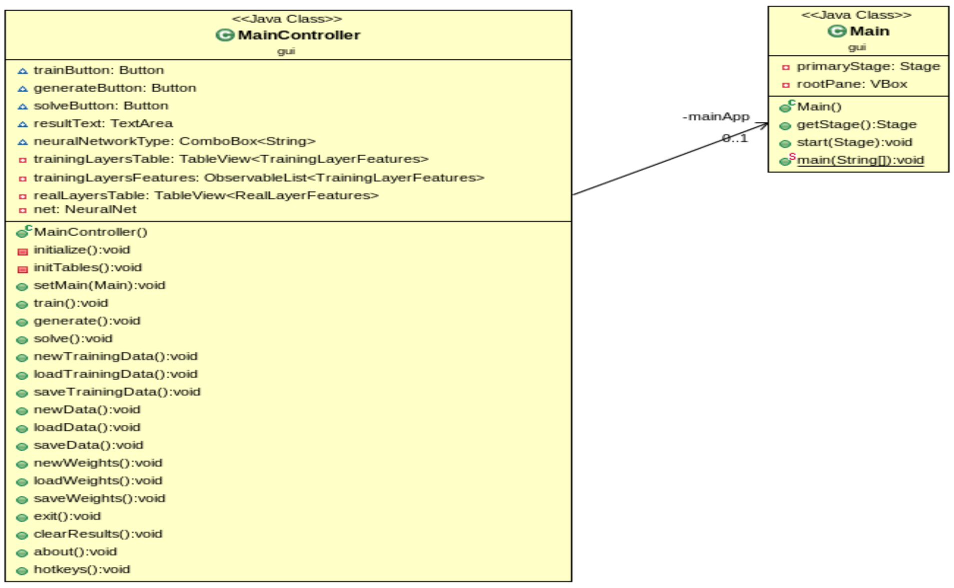


Figure 4: GUI package class diagram

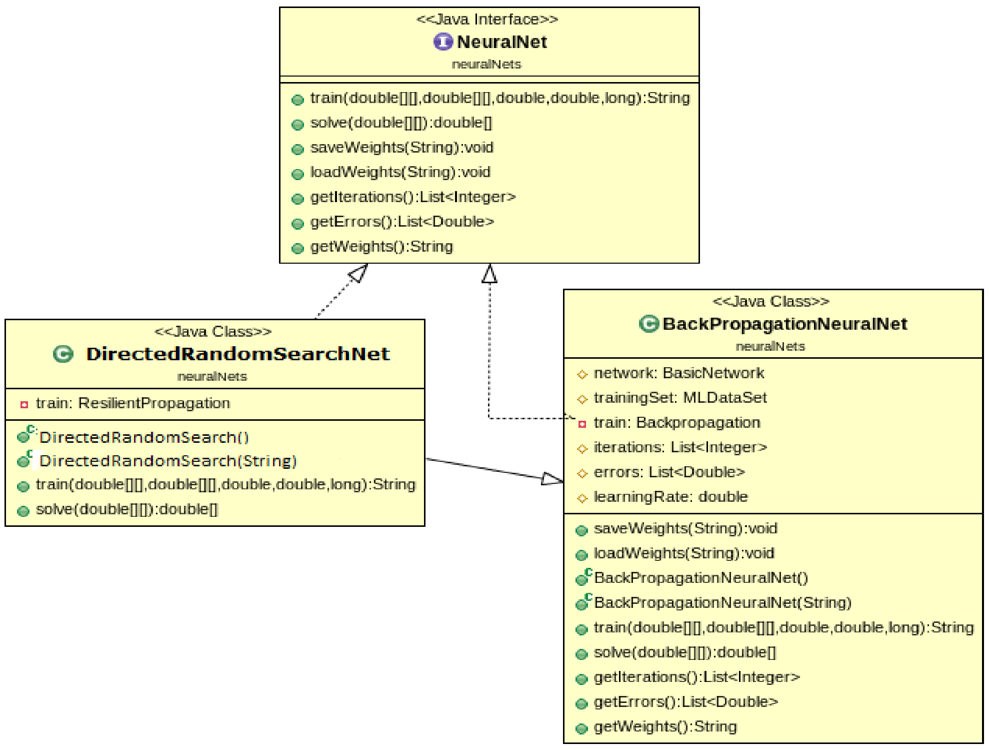


Figure 5: Neural nets class diagram

2.2. Description of neural network structure

To accomplish this task, a neural network (multilayer perceptron) with 3 layers was designed. The input layer consists of 4 neurons (due to the size of the input image), the hidden layer consists of 10 neurons (the size is chosen empirically) and the output layer contains one neuron (Fig. 6) [22].

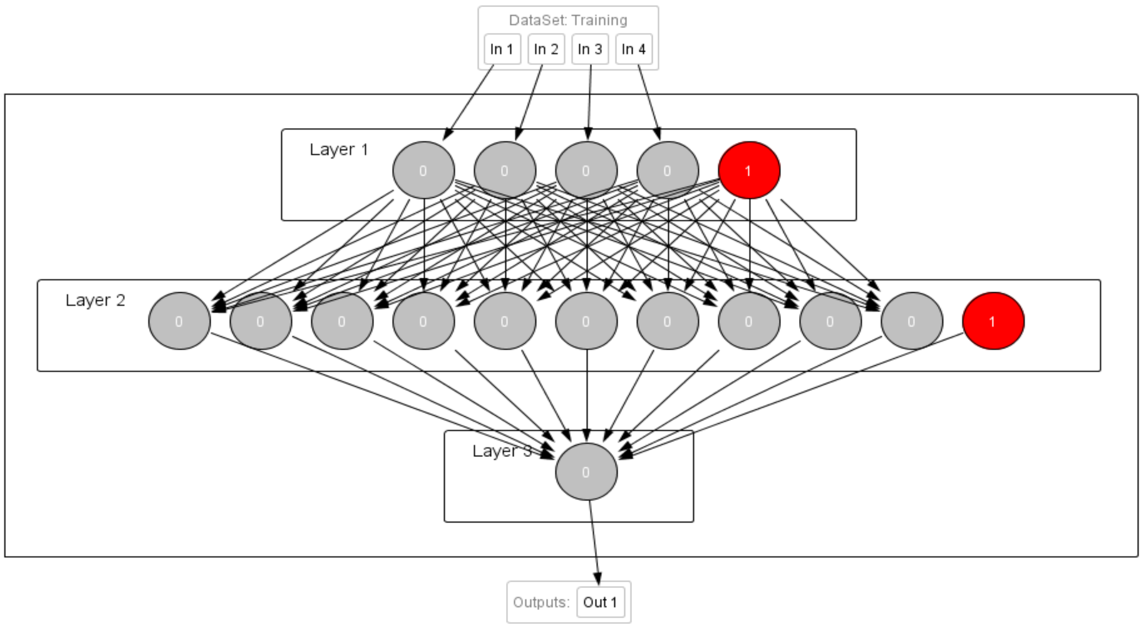


Figure 6: The structure of the neural network

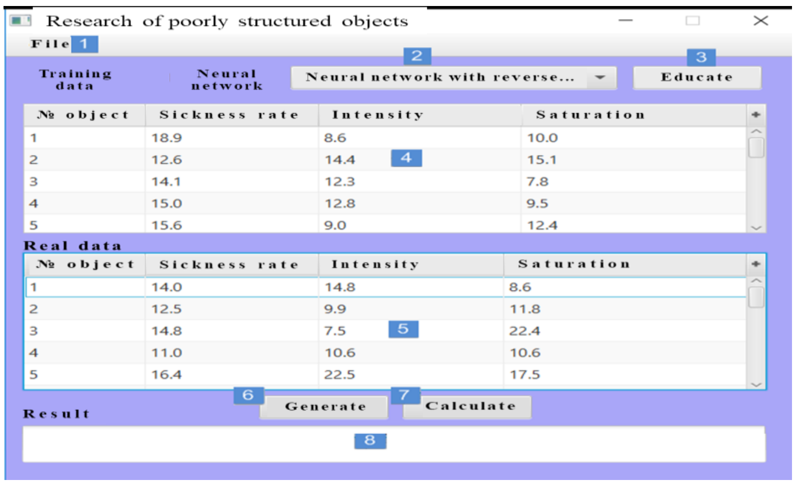
For the hidden layer the logistic unipolar function - sigmoid (11) is used.

$$f(z) = \frac{1}{1 + e^{-\alpha z}} \tag{10}$$

The value of landslide neurons - units. Initial values of weights are small random numbers within [-0.3; 0.3].

2.3. GUI Description

The application is focused on cross-platform so developed by means of Java SE 14 and JavaFX (note, to run the application must be installed JRE 10.0.2). To get acquainted with the functionality of the application, consider the main window of the program (Fig. 7).



№ object	Sickness rate	Intensity	Saturation
1	18.9	8.6	10.0
2	12.6	14.4	15.1
3	14.1	12.3	7.8
4	15.0	12.8	9.5
5	15.6	9.0	12.4

Real data

№ object	Sickness rate	Intensity	Saturation
1	14.0	14.8	8.6
2	12.5	9.9	11.8
3	14.8	7.5	22.4
4	11.0	10.6	10.6
5	16.4	22.5	17.5

Figure 7: Main program window

1 – Menu file
2 – Selection of neural network type
3 – Button to start the training of the neural network
4 – Training data table
5 – Table of forecasted data
6 – Button for generation of synthetic data
7 – The button to start the process of classifying objects by the neural network
8 – Text field of execution results

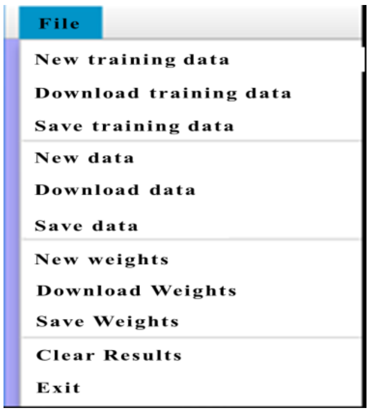
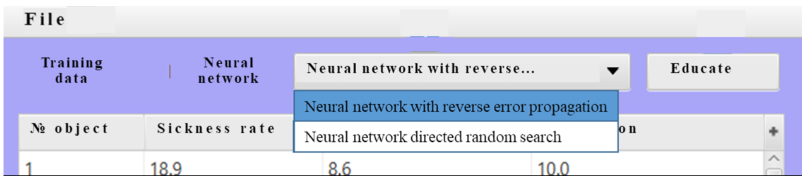


Figure 8: Menu file

The file menu includes the following sub-items:
1 - (New training data) - clearing the "Training data" table;
2 - (Download training data) - uploading data to the "Training data" table from the CSV file;
3 - (Save training data) - save data from "Training data" table to CSV-file;
4 - (New data) - clearing the "Real data" table;
5 - (Download data) - upload data to the table "Real data" from the CSV-file;
6 - (Save data) - save data from table "Real data" to CSV-file;
7 - (New weights) - clear weights for neural network;
8 - (Download Weights) - download weights from the file;
9 - (Save Weights) - save weights to a file;
10 - (Clear Results) - clear the text field results;
11 - (Exit) - finish the program;

To increase the functionality of the application, dynamic tables were implemented, i.e. you can add new records to the table, edit and delete existing ones, as well as upload and save unstructured and poorly structured medical data to a file. To improve the functionality of the application, dynamic tables were implemented, i.e. you can add new records to the table, edit and delete existing ones, as well as upload and save data to a file [28, 35]. To load data into a table, you need to go to the menu "File" -> ("Download training data" or "Download data") and select the file you want to upload (only CSV files are supported). The data and weights are saved in the same way - "File" -> ("Save Training Data", "Save Data" or "Save Weights" respectively).

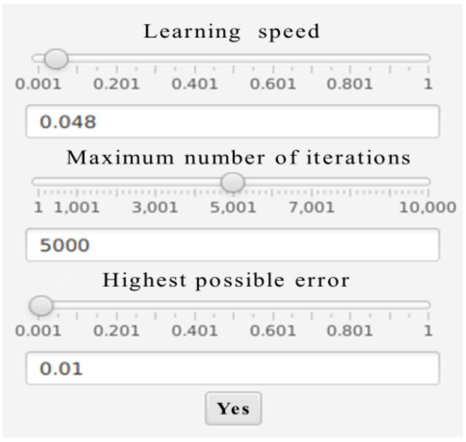
To train a neural network, first you need to select the algorithm by which it will learn (Fig. 9).


File
Training data
Neural network
Neural network with reverse...
Educate
Neural network with reverse error propagation
Neural network directed random search
№ object
Sickness rate
on

№ object	Sickness rate		
1	18.9	8.6	10.0

Figure 9: Setting of neural network training parameters

After that press the "Train" button and set the neural network training parameters, such as: training speed, maximum number of iterations and maximum permissible error (Fig. 10).


Learning speed

0.048

Maximum number of iterations

5000

Highest possible error

0.01

Figure 10: Setting of neural network training parameters

After setting the training parameters for the neural network, you need to press the "Yes" key to start training. When the neural network finishes training, the user will see the graph of error change (Fig. 11, Fig. 12), as well as information about training results (Fig. 13) and final weights, which can be saved to a file.

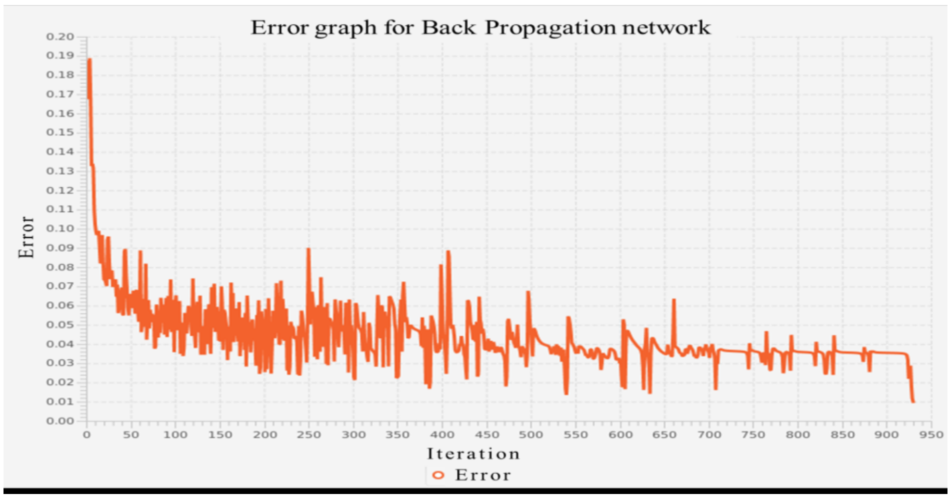


Figure 11: Error change schedule for Back Propagation network

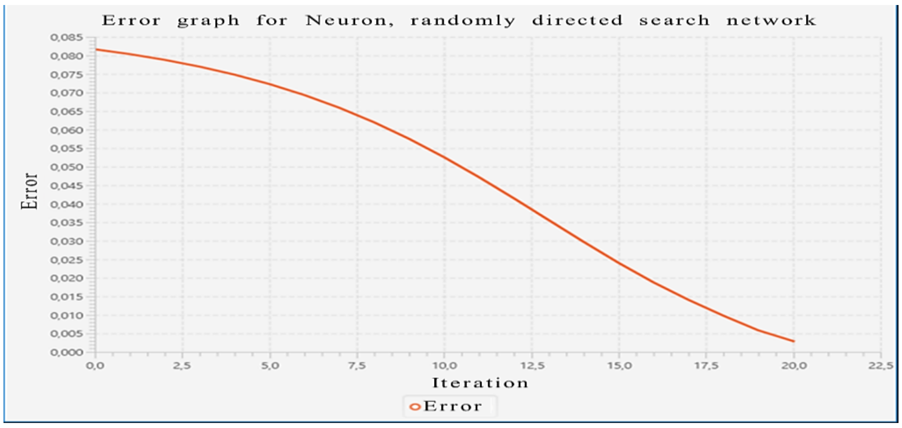
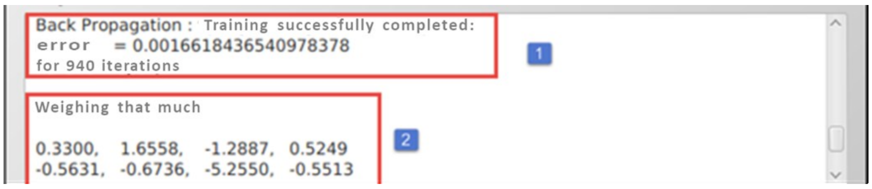


Figure 12: Error change schedule for Directed Random Search network (randomly directed search)



Back Propagation : Training successfully completed:
error = 0.0016618436540978378
for 940 iterations

Weighing that much

0.3300, 1.6558, -1.2887, 0.5249
-0.5631, -0.6736, -5.2550, -0.5513

Figure 13: Learning outcomes

According to the error graphs (Fig. 11, Fig. 12) and the number of iterations, it can be concluded that the Directed Random Search learning algorithm, although more complex to implement, works much faster than the classic Back Propagation. With the same error and speed of learning, training the network using Directed Random Search completed in the twentieth iteration, and using Back Propagation for nine hundred and forty.

Synthetic analytical processing of unstructured and poorly structured medical data generated by the "3 sigma" rule is used to test the neural network operation. To generate artificial medical data, press the "Generate" key, specify the number of columns to be filled with artificial medical data and press the "Yes" key (Fig. 14).

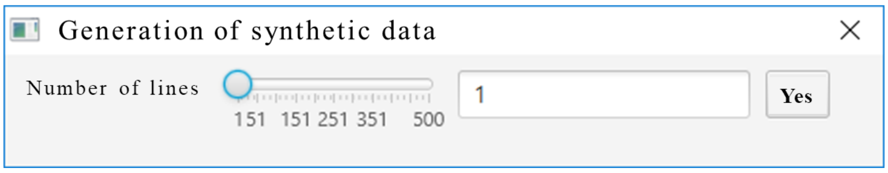
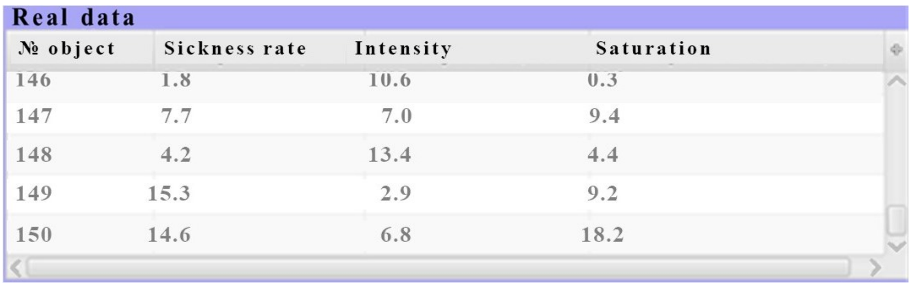


Figure 14: Options for generation of synthetic data

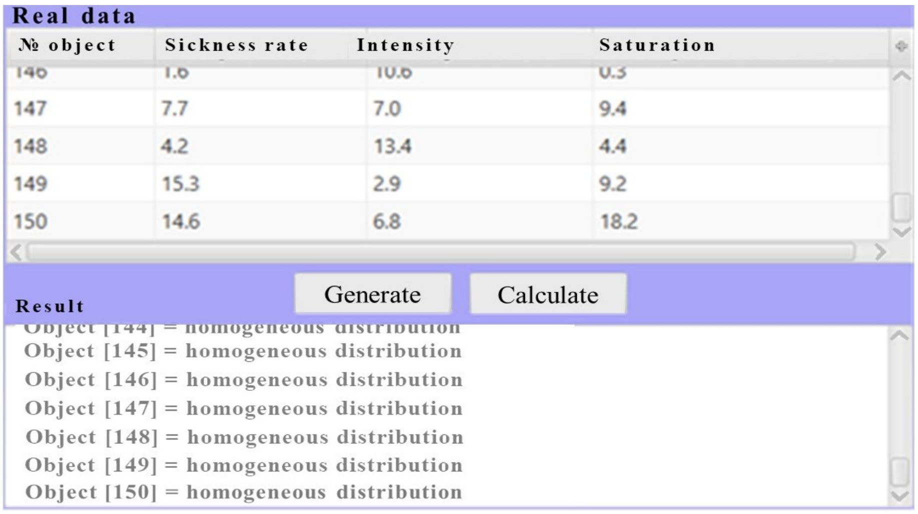
After the generation process is complete, the table will be filled with the corresponding number of columns with artificial data (Fig. 15, Fig. 16).



Real data

№ object	Sickness rate	Intensity	Saturation
146	1.8	10.6	0.3
147	7.7	7.0	9.4
148	4.2	13.4	4.4
149	15.3	2.9	9.2
150	14.6	6.8	18.2

Figure 15: The result of generating synthetic data



Real data

№ object	Sickness rate	Intensity	Saturation
146	1.6	10.6	0.3
147	7.7	7.0	9.4
148	4.2	13.4	4.4
149	15.3	2.9	9.2
150	14.6	6.8	18.2

Result

```
Object [144] = homogeneous distribution
Object [145] = homogeneous distribution
Object [146] = homogeneous distribution
Object [147] = homogeneous distribution
Object [148] = homogeneous distribution
Object [149] = homogeneous distribution
Object [150] = homogeneous distribution
```

Figure 16: Calculation result

Conclusions

The main objective of this work was to review the problem of neural networks and their applications, especially for management and control. A software product implementing the neural network was developed and learned from user-defined medical data. After the training, the program provides support for decision-making based on what it has learned. This program is focused on cross-platform, so it is made by means of Java SE14, and the graphical interface is designed by means of Java FX. The software product implements such a neural network error reverse propagation network (BackPropagation) and the network of directed random search (Directed Random Search). The neural network is trained and further recognizes the type of distribution (uniform, normal) by the specified characteristics. The "3 Sigma" rule is used for generation of synthetic unstructured and poorly structured medical data. According to the research we can conclude that the learning algorithm Directed Random Search, although there is more difficult to implement, but works much faster than the classic Back Propagation. With the same error and speed of learning, training the network using Directed Random Search can be several times faster than Back Propagation.

References

[1] A. Lisovskaya and T. Skripnik, "Processing of Neural System Information with the Use of Artificial Spiking Neural Networks," 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), Saint Petersburg and Moscow, Russia, 2019, pp. 1183-1186, doi: 10.1109/EIConRus.2019.8656651.

[2] A. Liu, Y. Yang, Q. Sun and Q. Xu, "A Deep Fully Convolution Neural Network for Semantic Segmentation Based on Adaptive Feature Fusion," 2018 5th International Conference on Information Science and Control Engineering (ICISCE), Zhengzhou, 2018, pp. 16-20, doi: 10.1109/ICISCE.2018.00013.

[3] Andrunyk, V., Vasevych, A., Chyrun, L., Chernovol, N., Antonyuk, N., Gozhyj, A., Gozhyj, V., Kalinina, I. and Korobchynskyi, M. (2020). Development of information system for aggregation and ranking of news taking into account the user needs. Paper presented at the *CEUR Workshop Proceedings, , 2604* 1127-1171.

[4] B. J. Isaac, H. Kinjo, K. Nakazono and N. Oshiro, "Suitable Activity Function of Neural Networks for Data Enlargement," 2018 18th International Conference on Control, Automation and Systems (ICCAS), Daegwallyeong, 2018, pp. 392-397.

[5] D. Ageyev, A. Mohsin, T. Radivilova and L. Kirichenko, "Infocommunication Networks Design with Self-Similar Traffic," *IEEE 15th International Conference on the Experience of Designing and Application of CAD Systems (CADSM)*, Polyana, Ukraine, 2019, pp. 24-27, doi: 10.1109/CADSM.2019.8779314.

[6] F. Lotfi, V. Ajallooeian and H. D. Taghirad, "Robust Object Tracking Based on Recurrent Neural Networks," 2018 6th RSI International Conference on Robotics and Mechatronics (IcRoM), Tehran, Iran, 2018, pp. 507-511, doi: 10.1109/ICRoM.2018.8657608.

[7] G. Zhou, L. Lv, X. Qiao and L. Jin, "Hierarchical Attention-based Fuzzy Neural Network for Subject Classification of Power Customer Service Work Orders," 2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), New Orleans, LA, USA, 2019, pp. 1-6, doi: 10.1109/FUZZ-IEEE.2019.8858852.

[8] Hengliang Tang, Yuan Mi, Fei Xue, Yang Cao, "An Integration Model Based on Graph Convolutional Network for Text Classification", *Access IEEE*, vol. 8, pp. 148865-148876, 2020.

[9] I. Dronyuk, O. Fedevych and N. Kryvinska, "High Quality Video Traffic Ateb-Forecasting and Fuzzy Logic Management," 2019 7th International Conference on Future Internet of Things and Cloud (FiCloud), Istanbul, Turkey, 2019, pp. 308-311, doi: 10.1109/FiCloud.2019.00051.

[10] I. Dronyuk, Y. Klishch and S. Chupakhina, "Developing Fuzzy Traffic Management for Telecommunication Network Services," 2019 IEEE 15th International Conference on the Experience of Designing and Application of CAD Systems (CADSM), Polyana, Ukraine, 2019, pp. 1-4, doi: 10.1109/CADSM.2019.8779323.

[11] J. C. Heck and F. M. Salem, "Simplified minimal gated unit variations for recurrent neural networks," 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), Boston, MA, 2017, pp. 1593-1596, doi: 10.1109/MWSCAS.2017.8053242.

[12] K. Vulinović, L. Ivković, J. Petrović, K. Skračić and P. Pale, "Neural Networks for File Fragment Classification," 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 2019, pp. 1194-1198, doi: 10.23919/MIPRO.2019.8756878.

[13] M. Benyamini, S. R. Nason, C. A. Chestek and M. Zacksenhouse, "Neural Correlates of error processing during grasping with invasive brain-machine interfaces*," 2019 9th International IEEE/EMBS Conference on Neural Engineering (NER), San Francisco, CA, USA, 2019, pp. 215-218, doi: 10.1109/NER.2019.8717020.

[14] M. Pasyeka, V. Sheketa, N. Pasieka, S. Chupakhina and I. Dronyuk, "System Analysis of Caching Requests on Network Computing Nodes," 2019 3rd International Conference on Advanced Information and Communications Technologies (AICT), Lviv, Ukraine, 2019, pp. 216-222, doi: 10.1109/AIACT.2019.8847909.

[15] Medykovskyy, M., Pasyeka, M., Pasyeka, N. & Turchyn, O. (2017). Scientific research of life cycle perfomance of information technology. Paper presented at the *Proceedings of the 12th International Scientific and Technical Conference on Computer Sciences and Information Technologies, CSIT 2017, , 1* 425-428. doi:10.1109/STC-CSIT.2017.809882

[16] Mishchuk, O., & Tkachenko, R. (2019). One-step prediction of air pollution control parameters using neural-like structure based on geometric data transformations. Paper presented at the *2019 11th International Scientific and Practical Conference on Electronics and Information Technologies, ELIT 2019 - Proceedings,* 192-196. doi:10.1109/ELIT.2019.8892333

[17] Nazarkevych, M., Lotoshynska, N., Brytkovskyi, V., Dmytruk, S., Dordiak, V., & Pikh, I. (2019). Biometric identification system with ateb-gabor filtering. Paper presented at the *2019 11th International Scientific and Practical Conference on Electronics and Information Technologies, ELIT 2019 - Proceedings,* 15-18. doi:10.1109/ELIT.2019.8892282

[18] Ö. F. Ertuğrul, R. Tekin and Y. Kaya, "Randomized feed-forward artificial neural networks in estimating short-term power load of a small house: A case study," 2017 International Artificial Intelligence and Data Processing Symposium (IDAP), Malatya, 2017, pp. 1-5, doi: 10.1109/IDAP.2017.8090344.

[19] Pasieka, N., Sheketa, V., Romanyshyn, Y., Pasieka, M., Domska, U., & Struk, A. (2019). Models, methods and algorithms of web system architecture optimization. Paper presented at the *2019 IEEE International Scientific-Practical Conference: Problems of Infocommunications Science and Technology, PIC S and T 2019 - Proceedings,* 147-152. doi:10.1109/PICST47496.2019.9061539

[20] Pasyeka, M., Sheketa, V., Pasieka, N., Chupakhina, S., & Dronyuk, I. (2019). System analysis of caching requests on network computing nodes. Paper presented at the *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT 2019 - Proceedings,* 216-222. doi:10.1109/AIACT.2019.8847909

[21] Q. Wang and M. Iwaihara, "Deep Neural Architectures for Joint Named Entity Recognition and Disambiguation," 2019 IEEE International Conference on Big Data and Smart Computing (BigComp), Kyoto, Japan, 2019, pp. 1-4, doi: 10.1109/BIGCOMP.2019.8679233.

[22] R. Jozefowicz, W. Zaremba and I. Sutskever, "An empirical exploration of recurrent network architectures", Proc. Int'l Conf. on Machine Learning, pp. 2342-2350, 2015.

[23] R. Wang, Z. Li, J. Cao, T. Chen and L. Wang, "Convolutional Recurrent Neural Networks for Text Classification," 2019 International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 2019, pp. 1-6, doi: 10.1109/IJCNN.2019.8852406.

[24] Romanyshyn, Y., Sheketa, V., Pikh, V., Poteriailo, L., Kalambet, Y. & Pasieka, N. (2019). Social-communication web technologies in the higher education as means of knowledge transfer. Paper presented at the *IEEE 2019 14th International Scientific and Technical Conference on Computer Sciences and Information Technologies, CSIT 2019 - Proceedings, , 3* 35-38. doi:10.1109/STC-CSIT.2019.8929753

[25] S. Ying and Q. Jianguo, "A Method of Arc Priority Determination Based on Back-Propagation Neural Network," 2017 4th International Conference on Information Science and Control Engineering (ICISCE), Changsha, 2017, pp. 38-41, doi: 10.1109/ICISCE.2017.18.

[26] Seungwan Seo, Czangyeob Kim, Haedong Kim, Kyounghyun Mo, Pilsung Kang, "Comparative Study of Deep Learning-Based Sentiment Classification", *Access IEEE*, vol. 8, pp. 6861-6875, 2020.

[27] T. Dong and T. Huang, "Neural Cryptography Based on Complex-Valued Neural Network," in 2019 IEEE Transactions on Neural Networks and Learning Systems, doi: 10.1109/TNNLS.2019.2955165.

[28] Tao Dong, Qinqin Zhang, "Dynamics of a Hybrid Circuit System With Lossless Transmission Line", *Access IEEE*, vol. 8, pp. 92969-92976, 2020.

[29] Tianyu Gao, Jin Yang, Wenjun Peng, Luyu Jiang, Yihao Sun, Fangchuan Li, "A Content-Based Method for Sybil Detection in Online Social Networks via Deep Learning", *Access IEEE*, vol. 8, pp. 38753-38766, 2020.

[30] Ting He, Ying Liu, Chengyi Xu, Xiaolin Zhou, Zhongkang Hu, Jianan Fan, "A Fully Convolutional Neural Network for Wood Defect Location and Identification", *Access IEEE*, vol. 7, pp. 123453-123462, 2019.

[31] Tkachenko, R., Izonin, I., Kryvinska, N., Dronyuk, I., & Zub, K. (2020). An approach towards increasing prediction accuracy for the recovery of missing iot data based on the grnn-sgtm ensemble. *Sensors (Switzerland), 20*(9) doi:10.3390/s20092625

[32] Tkachenko, R., Izonin, I., Vitynskyi, P., Lotoshynska, N., & Pavlyuk, O. (2018). Development of the non-iterative supervised learning predictor based on the ito decomposition and sgtm neural-like structure for managing medical insurance costs. Data, 3(4) doi:10.3390/data3040046

[33] V. Sheketa, L. Poteriailo, Y. Romanyshyn, V. Pikh, M. Pasyeka and M. Chesanovskyy, "Case-Based Notations for Technological Problems Solving in the Knowledge-Based Environment," 2019 IEEE 14th International Conference on Computer Sciences and Information Technologies (CSIT), Lviv, Ukraine, 2019, pp. 10-14, doi: 10.1109/STC-CSIT.2019.8929784.

[34] W. Huang, S. Oh and W. Pedrycz, "Hybrid Fuzzy Wavelet Neural Networks Architecture Based on Polynomial Neural Networks and Fuzzy Set/Relation Inference-Based Wavelet Neurons," in IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 8, pp. 3452-3462, Aug. 2018, doi: 10.1109/TNNLS.2017.2729589.

[35] Xiangyu Bu, Tao Dong, "Differential Privacy Optimal Consensus for Multiagent System by Using Functional Perturbation", *Information Cybernetics and Computational Social Systems (ICCSS) 2019 6th International Conference on*, pp. 157-162, 2019.

[36] Y. Huang, L. F. Capretz and D. Ho, "Neural Network Models for Stock Selection Based on Fundamental Analysis," 2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE), Edmonton, AB, Canada, 2019, pp. 1-4, doi: 10.1109/CCECE.2019.8861550.

[37] Y. Lin, C. Chou, S. Yang, H. Lai, Y. Lo and Y. Chen, "Neural Decoding Forelimb Trajectory Using Evolutionary Neural Networks with Feedback-Error-Learning Schemes," 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Honolulu, HI, 2018, pp. 2539-2542, doi: 10.1109/EMBC.2018.8512775.

[38] Yan Cheng, Leibo Yao, Guoxiong Xiang, Guanghe Zhang, Tianwei Tang, Linhui Zhong, "Text Sentiment Orientation Analysis Based on Multi-Channel CNN and Bidirectional GRU With Attention Mechanism", *Access IEEE*, vol. 8, pp. 134964-134975, 2020.

[39] Z. Mohammadi, A. Klug, C. Liu and T. C. Lei, "Data reduction for real-time enhanced growing neural gas spike sorting with multiple recording channels," 2019 9th International IEEE/EMBS Conference on Neural Engineering (NER), San Francisco, CA, USA, 2019, pp. 1084-1087, doi: 10.1109/NER.2019.8717062.

[40] Z. Ying, Z. Xing, C. Jian and S. Hui, "Processor Free Time Forecasting Based on Convolutional Neural Network," 2018 37th Chinese Control Conference (CCC), Wuhan, 2018, pp. 9331-9336, doi: 10.23919/ChiCC.2018.8483132.