Neural Network Technologies for Diagnosing Heart Disease

Ievgen Sidenko^a, Galyna Kondratenko^a and Valentyn Petrovych^a

^aPetro Mohyla Black Sea National University, 68th Desantnykiv Str., 10, Mykolaiv, 54003, Ukraine

Abstract

This paper describes the use of artificial neural networks to diagnose heart disease. Training and validation take place on an open data set that consists of 4 databases: Cleveland, Hungary, Switzerland, and Long Beach V. The complete data set consists of 76 attributes, but only 13 of them have an impact on heart disease. To solve the problem, 4 different neural networks are used. The first network was written in Kotlin, it is implemented without using third-party libraries, using only the simplest language constructs and the most popular mathematical methods that are used in neural networks. Networks were also created in Java with the DeepLearning4j library for deep learning, in Python with the Keras Tensor-Flow library, and the Matlab system. All networks were tested for training accuracy and the number of difficult cases (network confidence as a result). The results of the networks were checked and compared. The trained networks can be used both in embedded and mobile systems and in desktop systems.

Keywords

Neural network, heart disease, artificial intelligence, machine learning, diagnosing, medicine, decision making

1. Introduction

Since the creation of the first computer to modern supercomputers, the main purpose of computers has been and remains to facilitate people's lives. What a person will calculate in hours, and possibly days, the processor can calculate in a few seconds. Thanks to this property, a person trusts the computer even the question of medicine [1].

There are downsides to automating medicine. The processor executes only the commands that are written to it. It can be wrong because the person who programmed it can be wrong. Besides, the mistake can worsen the patient's condition [1]. Undoubtedly, computers help in medicine, but they are only a means of decision support and the last word is always up to the doctor [2]. Computers cannot eliminate the need for a man.

The difficulties for a person in medicine are based on the fact that a person does not have unlimited memory and cannot process a large amount of information at once, unlike a computer. For example, the symptoms of some viruses may be similar, and the number of tests may be too large to immediately infer and distinguish viruses. Predicting disease and ensuring human

ICT&ES-2020: Information-Communication Technologies & Embedded Systems, November 12, 2020, Mykolaiv, Ukraine ivgen.sidenko@chmnu.edu.ua (I. Sidenko); halyna.kondratenko@chmnu.edu.ua (G. Kondratenko); valllentua9128@gmail.com (V. Petrovych)

 ^{0000-0001-6496-2469 (}I. Sidenko); 0000-0002-8446-5096 (G. Kondratenko); 0000-0003-2035-1396 (V. Petrovych)
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000

CEUR Workshop Proceedings (CEUR-WS.org)

health is the first and most important goal for every country. Therefore, the topic of medicine using intelligent systems [3, 4] has been and will be relevant for many years.

Computers are used in medicine in almost every field: diagnostics, treatment, research, or data management. Despite the importance of each process, the diagnosis has always played a leading role. Because without the definition of a specific disease, it is impossible to begin effective treatment. In today's world, thanks to the high computing power of personal computers, it is possible to tell with impressive accuracy whether a person is sick or not. Modern diagnostics consists of three main stages: obtaining a medical history, examining the patient in fact, and conducting laboratory tests, and all these stages involve the computing power of processors [2, 5].

2. Related works and problem statement

The task of this work is to create a system that can accurately determine the presence of heart disease for certain input parameters. An open database with 1.026 patient records will be used for this purpose. This data set consists of 76 characteristics, starting with the last name and ending with the chemical composition in human blood. However, due to the existence of data that do not affect heart disease (surname, city of residence, etc.), the entire database is not used for training. According to all published studies, only 13 of them should be used, only those that do have a biological effect on heart disease.

According to statistics [6], heart problems (compared to all other organs) cause the highest number of deaths, this statistic is similar for each country. Therefore, the most effective artificial neural networks are used in cardiology. As an example, there is the possibility of using networks to diagnose posthypoxic disorders of the cardiovascular system in newborns [7]. This was done by a group of scientists led by T.V. Bowls and co-authors. The input data was a recording of the child's 5-minute heart rate, but to transfer the entire recording to the ANN would not give satisfactory results. It was decided to represent the intervals between strokes in the form of a histogram. Two specialized 3-layer networks were created. The study was conducted by various methods, but the most effective was the method of the back-propagation algorithm. The authors trained 2 neural networks at once to diagnose ST T segment of the heart and myocardial dysfunction. They managed to achieve 91% and 82% accuracy, respectively.

A huge amount of research is being done and a lot of attention is being paid to the use of computers to diagnose coronary heart disease. This attention is caused by the fact that this disease is the most common cause of sudden death in the world [8].

The first to do the first research in this area was a group of scientists led by M.C. Kolak, they described this in their article [9]. Eight different learning algorithms were used to create artificial neural networks. All showed a high result of sensitivity, specificity, and accuracy over 71%. In contrast to the main goal, namely the diagnosis of the disease, the study of other teaching methods was also conducted and their effectiveness was proved on small data, in contrast to the method of reverse propagation of the error.

Another group of scientists led by O.Y. Atkov. In their work [10] continued to study this disease. They used not only artificial neural networks but also the method of reference vectors. The latter belongs to the class of linear classifiers and is used for classification problems.

Thanks to the data mining technology, they managed to obtain 1324 records of patients with 25 parameters using the data of 3 hospitals. After data normalization, SPSS [11, 12, 13] and R [14] were used for statistical calculations. According to their calculations and their database, the method of reference vectors has higher accuracy, sensitivity, and speed of operation than the method with artificial neural networks. Computer's diagnostics of cardiovascular diseases considered in papers [15, 16].

There is a study diagnosing coronary heart disease with coronary atherosclerosis [17]. During the work, the analysis of dependence between 32 parameters of medical analyzes by a method of pair correlation with a prediction of both diseases was carried out. After the study, a study of the most influential factors was conducted. Among all, only the 10 most influential remained. As a result of the work and adjustment of the neural network, an accuracy of 96% was achieved. This accuracy makes it possible to detect the disease as accurately as possible and only thanks to the results of general medical tests. Approximately the same result was achieved by scientists M. Niranjan Murthy and M. Meenakshi in their work [18], but they used only 13 parameters and not only tests but also heart rate, sex, type of veins. The result of their work is a neural network with an accuracy of 95.5%. The neural network had the architecture of 13 input, 13 hidden, and 1 output neuron.

A very interesting combination of two technologies of artificial intelligence was made by Z. Arabasadi [19]. The genetic algorithm and the artificial neural network were combined. Thanks to this hybrid, a 10% increase in the accuracy of the neural network were obtained. The genetic algorithm offered the best scales for the network. Thanks to this, 93.85% accuracy, 97% sensitivity, and 92% specificity were obtained.

Nonlinear control stability of a heart beat tracking system is investigated in paper [20] based on second-order original and third-order modified Zeeman's heartbeat models.

Deep neural network training is even used to detect coronary heart disease. This was done by A. Kaliskan and M.E. Yuksel in their system of deep learning [21]. The network studied on 2 data sets and reached 87.6% and 89.7%. 13 parameters were fed to the input. To obtain such accurate results, normalization was performed and 400 iterations were performed. Since there is not much data, each database was divided into two parts, 70% for training and 30% for verification.

3. Neural network technologies for solving the task

A database from four different countries is used. However, it consists of the same fields, so a complete merge is performed. After that, a random sorting is performed. This is done in order to properly separate the data for testing and training. Without random sorting, training will be carried out on data from three countries, and testing on data from another country. Moreover, there may be a deviation in the chemical composition of blood, cholesterol, etc., because in different countries people eat different foods that have different environments, which can have an impact on the composition of the blood. Therefore, to avoid such a deviation, a random sorting is used. Before the start of working with data, we need to divide it into 2 groups: for training and accuracy. In our case, 126 for testing and 900 for training.

One of the most flexible methods for solving a current problem is to create an implementation

of a neural network using simple mathematical operations. The Kotlin programming language is used because it is concise, fast, and can be run on any device [22, 23]. In addition, any other programming language can be used. The application works in console mode. During the development of the neural network, much attention is paid to the flexibility of the neural network for different input data and the ability to change the internal parameters for different tasks. To begin with, matrix operations were implemented. When creating the functionality of the neural network to initialize the initial weights used the formula:

$$w = gaussian \cdot n^{-0.5},\tag{1}$$

where *w* is the weight; *gaussian*() is the Gaussian function, normal distribution with center at zero and normal deviation; *w* is the number of inputs to this node.

This is one of the many methods of initializing weights. It has a more accurate initialization and shows better results [24].

When creating a neural network, you need to calculate the error of the network for it to adjust its weight. This is usually the difference between the desired value and actual result, but created network use:

$$Error = (Desiredvalue - Actualvalue)^2.$$
 (2)

The created network uses not an ordinary error, but an error square. This is very convenient because when the value of the error is less than 1, the result decreases $0.1^2 = 0.01$, and when the values are greater than 1, the result increases $10^2 = 100$. This method pays attention to large errors, and small errors have less impact.

To find the value of how much you need to change the weight for a particular node, use the formula:

$$\frac{\partial E}{\partial w_{jk}} = -(t_k - o_k) \cdot sigm(\sum w_{jk} \cdot o_j)(1 - sigm(\sum w_{jk} \cdot o_j)) \cdot o_j, \tag{3}$$

where $\frac{\partial E}{\partial w_{jk}}$ is the differential ratio, it means how much the value of the node error will change when the value of the weights changes; $(t_k - o_k) = e_j$ is the target value minus negative; $\sum w_{jk} \cdot o_j$ is the sum of all input signals smoothed by weights leading to node j; o_j is the output signal of the node.

The formula is used to update the weights themselves:

$$w_{jknew} = w_{jkold} - \alpha \frac{\partial E}{\partial w_{ik}},\tag{4}$$

where α is the coefficient of learning, it is the coefficient that reduces the magnitude of changes in order not to miss the desired result; w_{jkold} is the past weight.

The created neural network requires some parameters to work and then you can start learning [18]:

• The number of nodes for each layer is a parameter that is responsible for two characteristics: the number of layers and the number of nodes in each layer. For each task, you need to specify your number of nodes in the input layer (number of database parameters) and the output (network result). Also, the value for the hidden layer must be set according to the task. For our task we use a neural network of type 13-50-1, where 13 is the number of indicators of each patient, 1 is the final diagnosis, 50 is the number of neurons to find the interdependencies between the parameters;

- Learning speed is the value of data trust, this parameter affects the learning speed. Value 0.3 is used for our problem, which is explained by the small amount of data [9, 18];
- The activation function is a function that is used for each node to measure the output signal. In our case, the sigmoid activation function. It is most convenient for neural networks. Its feature is a decrease or increase of the output value depending on the input [2, 24];
- A random seed is a random value, relying on which the matrix of weights is filled. It is needed to study the different behavior of the network with different values of weights and the ability to reuse the most effective weights [18, 24].

All input data before use are normalized by the formula:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}},\tag{5}$$

where x_{min} and x_{max} are the smallest and largest value of the current parameter, respectively; x is the value to be normalized.

The next stage is learning. This uses the inverse propagation of the error. This is a function that takes two parameters. The first parameter is a matrix with input values that are fed to the input of the neural network. The second is the matrix of the desired output signal. Both parameters are transposed before starting work. All operations in this function are matrix functions, which are described in Section 2. That is, first the result of the network is obtained and compared with the correct one. According to the net error, the weights are adjusted.

Because the amount of data is quite small, the neural network needs to be trained more than once. One complete cycle of learning, which is based on all data, is called the era. More epochs lead to greater accuracy of results, but too much leads to retraining and as a result - the inability to correctly predict new data and learn in general [25].

The next stage is to check the accuracy of the training. To do this, read the data file for verification. After reading, we transmit data to the neural network. The result of the network is given by a single number in the range from 0 to 1, which when multiplied by 100 gives a percentage of confidence in the network that the person is sick. If the result of the neural network is greater than 0.6 (60%), the prediction indicates that the patient is ill. The value of 60% was chosen given that the database includes a lot of older people and characteristics that can be found even in healthy people. Therefore, it is necessary to be surer, whether the person is ill.

Together with this, the number of complex values (cases) is calculated - a value that shows how many predictions of the neural network is in the range of 35%–65%. This range has been chosen as questionable. That is, if the network has a result of about 0.55, it indicates that the person is not sick. However, maybe it just was not enough for a different result. The more such

Table 1	
Pairwise compai	rison of some network results and correct answers

Result/Set	1	2	3	4	5	6	7	8
Network result	0.99	0.00	0.93	0.00	0.04	0.98	0.00	0.98
Correct answer	1.00	0.00	1.00	0.00	0.00	1.00	0.00	1.00
Result/Set	9	10	11	12	13	14	15	16
Network result	1.00	0.00	0.03	0.04	0.05	0.01	0.96	0.27
Correct answer	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Result/Set	119	120	121	122	123	124	125	126
Network result	0.00	0.10	1.00	1.00	0.00	0.00	0.90	0.00
Correct answer	0.00	0.00	1.00	1.00	0.00	0.00	0.00	0.00

values, the less you can trust the network. Therefore, it is necessary to check the accuracy of a network not only on accuracy but also on the number of difficult values.

The training took place in 200 epochs. It took 14 seconds. Due to a large number of nodes in the hidden layer and the high learning rate, this was enough to get excellent results, namely 100% accuracy, and 0 complex cases. Generated random seeds -8666230825488870003.

Additionally, the correctness of the application was verified by a direct comparison of some network results and correct answers (Table 1).

According to the results, we can say that the neural network has learned to make very accurate predictions. If we take the data for verification, which were not included in the training sample, and transmit it to the neural network, we get the result in 100% correct answers and 0 complex cases. Such a number of complex cases mean the confidence of the neural network in their answers. As the results were very excellent, a full data set test was performed. The result was also 100% and only 8 complex cases, which of the total number (1025) corresponds to only 0.78% of the values. Based on this, you can confidently say about the excellent learning outcome and be confident in the results of the network.

DeepLearning4j is an open source library designed to implement neural networks in Java [26]. It can use not only CPU power but also video cards. It has a very flexible interface and can be used for most artificial intelligence tasks. Includes the ability not only to create neural networks, but also the preparation and normalization of data, fast structures for working with large data sets. One huge advantage is the huge number of built-in algorithms, activation functions, and data processing methods.

After reading, the data undergo normalization and transformation. The next stage is the configuration of the neural network. Specify the following parameters:

- Random seeds;
- The activation function is a sigmoid;
- The learning factor is 0.1, as higher values lead to retraining and reduce accuracy;

- The type of neural network is 13-50-1;
- Error propagation function is a gradient descent;
- The number of epochs is 200;
- Updater is Adam with parameter 0.1;
- The loss function is mean squared logarithmic error.

The result is an accuracy of 99.21% and 0 complex cases for 200 epochs and 100% accuracy for 230 epochs. Two hundred epochs passed in 1.89 seconds. The library uses the acceleration of calculations due to the power of the video card and processor.

TensorFlow is an open source platform for machine learning. It has a flexible ecosystem of tools, libraries, and resources [27, 28, 29]. Researchers can quickly add their developments to this platform, and programmers can easily use and integrate into any device. This platform can run on both a server and a local device. Each algorithm and its implementation can be easily viewed. The system provides convenient high-level APIs for working with complex networks.

At its core, TensorFlow has another open source library called Keras. It is a flexible and highperformance interface for solving machine learning problems. Much attention in the library is paid to deep learning. It provides the necessary abstractions and blocks for the development and implementation of high-bandwidth machine learning solutions. Like DeepLearning4j it can work using the power of graphics cores [21, 25, 26].

The data is read before starting work. This is realized due to the built-in function. The next step is to set up the network. Due to the peculiarities of working with neural layers in Keras, we create 4 layers. The first of which is the normalization layer.

Network settings:

- Batch normalization (axis = 1, momentum = 0.99, epsilon = 0.001);
- The activation function is a sigmoid;
- The optimizer is an Adam;
- Loss is a Mean Squared Error;
- Layers are 13-50-1.

Unlike previous networks, where there were enough 200 epochs, here 200 epochs give an accuracy of 85%. An attempt was made to increase the learning rate, but this led to retraining and results within 50%. Therefore, the only way out is to increase the number of epochs. Three hundred epochs show 95% accuracy, 400 - 97% accuracy, 500 - 100%. Based on this, the number of epochs was increased to 500.

The training lasted 22.5 seconds. But the network has learned to make predictions very well. Also, complex cases of only 2. Unlike previous networks, the result of this may be negative values and greater than 1.

Matlab is a system for scientists that combines a huge number of libraries for calculations in any field of science [25, 30]. The application perfectly combines mathematics, graphics, and programming. Unlike conventional programming languages, it also provides a graphical interface for many tasks. Many tasks are performed online, and the user can observe them and notice errors. Excellent optimization organizes extraordinary speed. But the minimum system needs are very high.

The distribution of data is done by example (Figure 1).

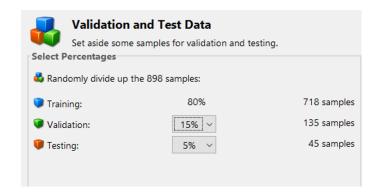


Figure 1: Distribution of data for training

This distribution was chosen based on the fact that only training data is used for training and not a complete database. The data for testing has been reduced to a minimum of 5%, according to the fact that we will evaluate the network ourselves. Also, the data for verification is left as a base 15%.

There are three basic methods of learning (Figure 2).

Train Network							
Choose a training algorithm:							
	Levenberg-Marquardt ~	•					
This algorithm typically	Levenberg-Marquardt	tine Testates					
	Bayesian Regularization	s time. Training ng, as indicated by an					
increase in the mean s	Scaled Conjugate Gradient	nples.					
Train using Levenberg-Marquardt. (trainlm)							

Figure 2: Algorithms of training in Matlab

The third method did not show good results. Its results are 88.1% accurate and 14 complex cases. Also, the first and second turned out to be very accurate. Both are 100% accurate and 0 complex cases. The first is the Levenberg-Marquardt algorithm. It took only 19 iterations (epochs) and 3 seconds to complete the training. The second is the Bayes regularization. It requires more time and iterations than the first one. It needs 82 iterations and 9 seconds.

Relying on the results of both algorithms, we can say with certainty that both algorithms are suitable for learning in this database. However, for faster processing and lower resource costs, it is better to use the Levenberg-Marquardt algorithm. It is about 3 times faster than Bayesian regularization.

4. Research and optimization of the created neural network on Kotlin

Consider the optimization of the developed neural network on Kotlin, because the corresponding implementation has more parameters for configuration and optimization in contrast to others [25, 27].

Simple creation of a neural network is not difficult, it is only working with matrices for calculations of nodes and implementation of a method of backpropagation of an error. However, this is not enough for efficient and accurate network operation. Therefore, after creating the implementation in the Kotlin language, the network setup was started.

After creating different implementations of neural networks, authors noticed different effects of the learning rate on network training. In some cases the value 0.5 is needed, in some it leads to retraining and the value 0.01 is enough. That is, they do not equally affect network learning. For example, as many as 0.3 was used for implementation on Kotlin, which is considered too high, but it did not lead to negative results, but only led to faster learning in fewer epochs. At the same time, DeepLearning4j uses 0.1, and when at least doubled, the network could not learn at all and everything came to retraining. TensorFlow with a Keras kernel was enough 0.001, for excellent training and as well as in DeepLearning4j led to retraining [25, 26, 27]. In Matlab, the learning factor is not specified because it is selected automatically by the system [30].

Retraining means the full study of certain data and the ability to distinguish only them, and other new ones can not predict. It was also observed during the wrong learning factor as follows: the neural network with each new epoch shows new values of accuracy, then greater, then less (80%, 75%, 82%, 73%, ...), this is an indicator that the network has a very large learning rate and with each era memorizes new data forgetting about the old. Since a lot depends on the initial initialization of weights, and they are generated by a random method, during the study and optimization of the network we will create not one network, but 30 for greater accuracy. Average, maximum, and minimum accuracy will be used as an indicator of the correctness of these networks [31, 32].

First of all, you need to explore the right number of epochs to learn from a database of patients. Learning takes place in steps of 20 epochs, i.e. from 1 epoch to 300 epochs 30 neural networks with different weights are created. Each network learns a step-dependent number of epochs. After that, its accuracy is checked. The next step is to calculate the average accuracy for all 30 networks. Other network parameters: learning speed is 0.3, the number of nodes on the hidden layer is 50. According to this data authors can build a graph (Figure 3).

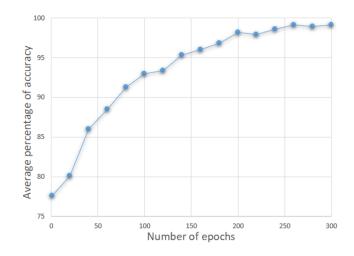


Figure 3: Dependence of learning accuracy on the number of epochs

According to the constructed graph (Figure 3), it is possible to investigate that with more epochs accuracy also increases. After 200 epochs you can get great results of accuracy.

The next stage of the study is the coefficient of learning speed (Figure 4). This ratio varies from 0 to 1. It affects the internal properties, such as prevention of retraining, speed of data learning, the level of confidence in the data. The study will be conducted in steps of 0.1 from 0 to 1. The number of epochs according to the previous study is 200. The number of nodes in the hidden layer is 50.

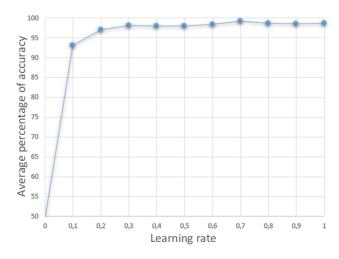


Figure 4: Dependence of learning accuracy on learning rate

According to the graphs, you can see that a value greater than 0.3 does not make sense at all. Smaller values give insufficient accuracy at 200 epochs, and larger ones give the same result and can make learning impossible. Therefore, it was decided to use exactly 0.3. According to the figure, even with the value of the coefficient of 0.3 neurons, the network can reach 100% accuracy.

The next step of the study is the number of nodes in hidden layer (Figure 5) [33, 34, 35]. Depending on this parameter, the elasticity of the network to search for interdependencies in the data increases, but also increases the complexity of calculations. This stage will be studied in steps of 10 from 1 to 100. The learning rate is 0.3, the number of epochs is 200.

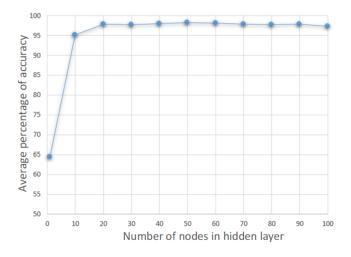


Figure 5: Dependence of learning accuracy on the number of nodes in hidden layer

The last stage of the study shows that the best result shows 50 nodes in hidden layer, these nodes are enough to find all the interdependencies in the data.

According to recent research, the best settings for network setup have been calculated. The following parameters are required for our database and network:

- The number of neurons in the hidden layer is 50;
- The number of epochs is 200;
- The learning rate is 0.3.

Using these parameters, a dozen neural networks are created and selected with the best accuracy parameters and the minimum number of complex cases.

5. Conclusions

During the writing of the work, the analysis of the use of artificial intelligence methods in medicine was carried out. The principles of work of neural network technologies are investigated. A statistical analysis of the patient database used for training was performed. Matrix operations for work with neural networks are implemented. The neural network is configured using the Kotlin programming language to diagnose heart disease. The settings of artificial intelligence

libraries to work with the same database have also been mastered. The analysis and comparison of different networks with the following characteristics: speed, convenience, and accuracy of training.

All four implementations showed the highest accuracy (100%), but at the same time there is a difference in the number of difficult cases, for example, when testing the TensorFlow implementation, the number of complex cases is 2, there is also a difference in the number of learning epochs (correspondingly time) for DeepLearning4j implementation, this value is 230 epochs (2.06 seconds).

Therefore, following the task, the networks were optimized for maximum accuracy, and comparisons were made on different characteristics. Each created application has an accuracy of 100%, so they can be used with complete confidence in the results.

The created systems can be integrated into any device. And thanks to this, it is possible to provide doctors only with software to check their patients. This will help to make an early diagnosis of the disease, and as a result, will reduce mortality from heart disease.

References

- [1] A. M. Yemelyanov, J. A. Berry, A. A. Yemelyanov, Decision support and error analysis of difficult decisions in clinical medicine, in: Advances in Neuroergonomics and Cognitive Engineering, Springer International Publishing, 2020, pp. 214–221. URL: https://doi.org/10. 1007/978-3-030-51041-1_29. doi:10.1007/978-3-030-51041-1_29.
- [2] E. Tànfani, A. Testi (Eds.), Advanced Decision Making Methods Applied to Health Care, Springer Milan, 2012. URL: https://doi.org/10.1007/978-88-470-2321-5. doi:10.1007/ 978-88-470-2321-5.
- [3] V. Sugumaran, Z. Xu, H. Zhou (Eds.), Application of Intelligent Systems in Multi-modal Information Analytics, Springer International Publishing, 2021. URL: https://doi.org/10. 1007/978-3-030-51431-0. doi:10.1007/978-3-030-51431-0.
- [4] P. Kushneryk, Y. P. Kondratenko, I. V. Sidenko, Intelligent dialogue system based on deep learning technology., in: ICTERI PhD Symposium, 2019, pp. 53–62.
- [5] U. Kose, O. Deperlioglu, J. Alzubi, B. Patrut, Diagnosing parkinson by using deep autoencoder neural network, in: Deep Learning for Medical Decision Support Systems, Springer Singapore, 2020, pp. 73–93. URL: https://doi.org/10.1007/978-981-15-6325-6_5. doi:10.1007/978-981-15-6325-6_5.
- [6] National center for chronic disease prevention and health promotion, division for heart disease and stroke prevention. centers for disease control and prevention. heart disease facts, 2020. URL: https://www.cdc.gov/heartdisease/facts.htm.
- [7] T. V. Chasha, N. V. Kharlamova, O. I. Klimova, F. N. Yasinsky, I. F. Yasinsky, The use of neural networks for predicting the course of post-hypoxic disorders of the cardiovascular system in newborn children, Bulletin of Ivanovo State Power Engineering University 4 (2009) 1–3.
- [8] Coronary heart disease, 2020. URL: https://prooneday.ru/zdorov-ja/hvorobi/ 409-ishemichna-hvoroba-sercja.html.
- [9] M. C. Çolak, C. Çolak, H. Kocatürk, S. Sagiroglu, İ. Barutçu, Predicting coronary artery

disease using different artificial neural network models/koroner arter hastaliginin degisik yapay sinir agi modelleri ile tahmini, Anadulu Kardiyoloji Dergisi: AKD 8 (2008) 249–254.

- [10] O. Y. Atkov, S. G. Gorokhova, A. G. Sboev, E. V. Generozov, E. V. Muraseyeva, S. Y. Moroshkina, N. N. Cherniy, Coronary heart disease diagnosis by artificial neural networks including genetic polymorphisms and clinical parameters, Journal of Cardiology 59 (2012) 190–194. URL: https://doi.org/10.1016/j.jjcc.2011.11.005. doi:10.1016/j.jjcc.2011.11.005.
- [11] Ibm spss statistics: Features and modules ibm., 2020. URL: https://www.ibm.com/products/ spss-statistics/details.
- [12] O. Martynyuk, A. Sugak, D. Martynyuk, O. Drozd, Evolutionary network model of testing of the distributed information systems, in: 2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), IEEE, 2017. URL: https://doi.org/10.1109/idaacs.2017.8095215. doi:10.1109/idaacs.2017.8095215.
- [13] S. McDonald, R. Vieira, D. W. Johnston, Analysing n-of-1 observational data in health psychology and behavioural medicine: a 10-step SPSS tutorial for beginners, Health Psychology and Behavioral Medicine 8 (2020) 32–54. URL: https://doi.org/10.1080/21642850. 2019.1711096. doi:10.1080/21642850.2019.1711096.
- [14] The r language for statistical computing, 2020. URL: https://www.r-project.org/.
- [15] I. P. Atamanyuk, Y. P. Kondratenko, Calculation method for a computer's diagnostics of cardiovascular diseases based on canonical decompositions of random sequences, in: CEUR Workshop Proceedings, 2015, pp. 108–120.
- [16] V. Shebanin, I. Atamanyuk, Y. Kondratenko, Y. Volosyuk, Canonical mathematical model and information technology for cardio-vascular diseases diagnostics, in: 2017 14th International Conference The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM), IEEE, 2017. URL: https://doi.org/10.1109/cadsm.2017.7916170. doi:10.1109/cadsm.2017.7916170.
- [17] A. G. Sboev, S. G. Gorokhova, N. N. Cherniy, Development of a neural network technique for early diagnosis of ischemic heart disease and coronary atherosclerosis, Bulletin of Voronezh State University 2 (2011) 204–213.
- [18] H. N. Murthy, M. Meenakshi, Ann model to predict coronary heart disease based on risk factors, Bonfring International Journal of Man Machine Interface 3 (2013) 13–18.
- [19] Z. Arabasadi, R. Alizadehsani, M. Roshanzamir, H. Moosaei, A. A. Yarifard, Computer aided decision making for heart disease detection using hybrid neural network-genetic algorithm, Computer Methods and Programs in Biomedicine 141 (2017) 19–26. URL: https://doi.org/10.1016/j.cmpb.2017.01.004. doi:10.1016/j.cmpb.2017.01.004.
- [20] M. Abdelhady, Y. Kondratenko, W. Abouelwafa, D. Simon, Stability analysis of heartbeat control based on the zeeman framework, in: 2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), IEEE, 2019. URL: https://doi.org/10.1109/idaacs.2019.8924412. doi:10.1109/idaacs.2019.8924412.
- [21] A. Caliskan, M. E. Yuksel, Classification of coronary artery disease data sets by using a deep neural network, The EuroBiotech Journal 1 (2017) 271–277. URL: https://doi.org/10. 24190/issn2564-615x/2017/04.03. doi:10.24190/issn2564-615x/2017/04.03.
- [22] L. Ardito, R. Coppola, G. Malnati, M. Torchiano, Effectiveness of kotlin vs. java in android

app development tasks, Information and Software Technology 127 (2020) 106374. URL: https://doi.org/10.1016/j.infsof.2020.106374. doi:10.1016/j.infsof.2020.106374.

- [23] D. Gotseva, Y. Tomov, P. Danov, Comparative study java vs kotlin, in: 2019 27th National Conference with International Participation (TELECOM), IEEE, 2019. URL: https://doi.org/ 10.1109/telecom48729.2019.8994896. doi:10.1109/telecom48729.2019.8994896.
- [24] S. Ansari, Building Computer Vision Applications Using Artificial Neural Networks, Apress, 2020. URL: https://doi.org/10.1007/978-1-4842-5887-3. doi:10.1007/ 978-1-4842-5887-3.
- [25] H. Iba, N. Noman (Eds.), Deep Neural Evolution, Springer Singapore, 2020. URL: https: //doi.org/10.1007/978-981-15-3685-4. doi:10.1007/978-981-15-3685-4.
- [26] S. Lang, F. Bravo-Marquez, C. Beckham, M. Hall, E. Frank, WekaDeeplearning4j: A deep learning package for weka based on deeplearning4j, Knowledge-Based Systems 178 (2019) 48–50. URL: https://doi.org/10.1016/j.knosys.2019.04.013. doi:10.1016/j.knosys.2019. 04.013.
- [27] M. Ashraf, S. M. Ahmad, N. A. Ganai, R. A. Shah, M. Zaman, S. A. Khan, A. A. Shah, Prediction of cardiovascular disease through cutting-edge deep learning technologies: An empirical study based on TENSORFLOW, PYTORCH and KERAS, in: Advances in Intelligent Systems and Computing, Springer Singapore, 2020, pp. 239–255. URL: https: //doi.org/10.1007/978-981-15-5113-0_18. doi:10.1007/978-981-15-5113-0_18.
- [28] R. Leizerovych, G. Kondratenko, I. Sidenko, Y. Kondratenko, IoT-complex for monitoring and analysis of motor highway condition using artificial neural networks, in: 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), IEEE, 2020. URL: https://doi.org/10.1109/dessert50317.2020.9125004. doi:10. 1109/dessert50317.2020.9125004.
- [29] I. Sidenko, G. Kondratenko, P. Kushneryk, Y. Kondratenko, Peculiarities of human machine interaction for synthesis of the intelligent dialogue chatbot, in: 2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), IEEE, 2019. URL: https://doi.org/10.1109/idaacs.2019.8924268. doi:10.1109/idaacs.2019.8924268.
- [30] D. Ballabio, M. Vasighi, A MATLAB toolbox for self organizing maps and supervised neural network learning strategies, Chemometrics and Intelligent Laboratory Systems 118 (2012) 24–32. URL: https://doi.org/10.1016/j.chemolab.2012.07.005. doi:10.1016/j. chemolab.2012.07.005.
- [31] V. Zinchenko, G. Kondratenko, I. Sidenko, Y. Kondratenko, Computer vision in control and optimization of road traffic, in: 2020 IEEE Third International Conference on Data Stream Mining & Processing (DSMP), IEEE, 2020. URL: https://doi.org/10.1109/dsmp47368. 2020.9204329. doi:10.1109/dsmp47368.2020.9204329.
- [32] M. Marefat, A. Juneja, Serverless data parallelization for training and retraining of deep learning architecture in patient-specific arrhythmia detection, in: 2019 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI), IEEE, 2019. URL: https://doi.org/10.1109/bhi.2019.8834566. doi:10.1109/bhi.2019.8834566.
- [33] G.-W. Cai, Z. Fang, Y.-F. Chen, Estimating the number of hidden nodes of the singlehidden-layer feedforward neural networks, in: 2019 15th International Conference on Computational Intelligence and Security (CIS), IEEE, 2019. URL: https://doi.org/10.1109/

cis.2019.00044. doi:10.1109/cis.2019.00044.

- [34] A. J. Thomas, M. Petridis, S. D. Walters, S. M. Gheytassi, R. E. Morgan, On predicting the optimal number of hidden nodes, in: 2015 International Conference on Computational Science and Computational Intelligence (CSCI), IEEE, 2015. URL: https://doi.org/10.1109/ csci.2015.33. doi:10.1109/csci.2015.33.
- [35] K.-Y. Huang, L.-C. Shen, J.-D. You, L.-S. Weng, Proof of hidden node number and experiments on RBF network for well log data inversion, in: 2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), IEEE, 2016. URL: https: //doi.org/10.1109/igarss.2016.7729721. doi:10.1109/igarss.2016.7729721.