Intelligent Neural Networks Models for the Technological Separation Processes

Andrey Kupin[a], Dmytro Zubov[b], Yuriy Osadchuk[a], Rodion Ivchenko[a] and Vadym Saiapin[a]

[a]*Kryvyi Rih National University, Ukraine, 50027, Kryvyi Rih, Vitaly Matusevich, 11*
[b]*University of Central Asia, Naryn, Kyrgyzstan, 722918, Naryn, Lenin Street, 310*

Abstract

A typical separation process is formalized using the example of iron ore (magnetite quartzites) beneficiation technology. This study examines the suitability of various neural network structures as mathematical models for regression. The results of computer modeling for training using real indicators of magnetite quartzite beneficiation and reference models are presented. Comparison of the approximation results for different neural network bases is also included. The authors tested various samples of reference and noisy data. The best intellectual models are recommended for automating separation processes.

Keywords [1]

Technological processes, beneficiation, magnetite quartzite, intelligent neural networks models.

1. Introduction

This study solving the problem of selecting optimal scientific approaches for formalizing technological processes (TP) in separation technology, with the aim of automating control. A typical example is the TP for beneficiation (separation or concentration) of iron ore (magnetite quartzites). Numerous works of the authors [1-3] have convincingly demonstrated that artificial intelligence (AI) technologies hold the greatest potential here. This is mainly due to factors such as non-linearity, non-stationarity, a large number of parameters, incomplete information, etc. [4] Despite the existence of numerous studies in this area, some issues require ongoing clarification and adaptation to the conditions of specific TP. These include the selection of architecture, topology, teaching methods, and hardware and software implementation, etc. [4-6] Therefore, this paper examines several promising intellectual approaches and mathematical models based on the use of artificial neural networks.

2. Formalization models of the process based on neural networks

A scheme is proposed [2] for using artificial intelligence technologies (neural networks, fuzzy logic, evolution, synergetic, etc.) to build models of technological processes. This scheme represents a formalized version of a multi-stage concentrating process.

The following notations are used in the scheme: Pi represents an apparatus or stage in the technological process (scheme PG); X represents the set of input actions that affect the control object. This set includes both external PG inputs to the process and inputs of individual devices Pi. The inputs can be controlled influences (i.e., control inputs) and uncontrolled influences (i.e., disturbances).

IntelITSIS'2023: 4th International Workshop on Intelligent Information Technologies and Systems of Information Security, March 22–24, 2023, Khmelnytskyi, Ukraine
EMAIL: kupin.andrew@gmail.com (A. Kupin); dzubovua@mail.ru (D. Zubov); u.osadchuk@knu.edu.ua (Y. Osadchuk); ivchenko.ra@gmail.com (R. Ivchenko); ksn.vadim@knu.edu.ua (V. Saiapin)
ORCID: 0000-0001-7569-1721 (A. Kupin); 0000-0002-5601-7827 (D. Zubov); 0000-0001-6110-9534 (Y. Osadchuk); 0000-0003-4252-4825 (R. Ivchenko); 0000-0002-7415-5158 (V. Saiapin)

Disturbances are sometimes measurable, but in other cases, they can only be assessed verbally (e.g., large, small, etc.). Y represents the set of outputs, which, similar to X, combines the outputs of the entire process and individual devices Pi. While all outputs are potentially controllable, in practice, only a subset of them are typically controlled. Not all connections between input signals and outputs can be determined using classical transfer functions (W(p)). This limitation is due to the dimension of the problem and the level of knowledge of the object under study. Although certain parameters (factors) may be known to exist, it may not always be possible to measure or evaluate them accurately enough. However, if a parameter is expected to have a significant effect on the dynamics of the process, it should be included in the consideration. If an accurate (quantitative) assessment is impossible, a fuzzy (linguistic) assessment can be assigned to a parameter. The implemented approach for developing technology for operational forecasting of concentration processes assumes the presence of implicit mutual influence among the technological parameters of the process and the characteristics of separation products.

To begin the analysis, we consider a scenario where the characteristics of the feedstock and all technological parameters are assumed to have a significant influence on the process outputs. The process model is represented as a directed graph, with the parameters as nodes and the arcs indicating their mutual influence. This representation of the technological process is similar to a neural network, where the nodes contain functions that convert signals.

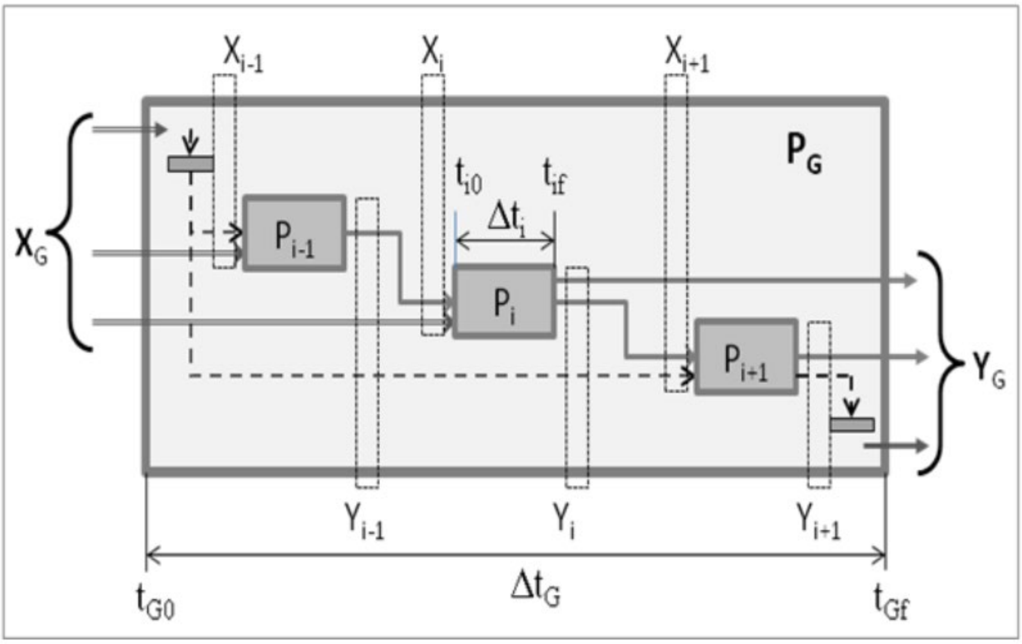


Figure 1: Scheme of a formal representation of a multi-stage technological process [2]

It is known that to formalize the beneficiation TP under the conditions of a technological line, we need to consider a number of parameters that can be represented as a "black box" in classical cybernetics (see Fig. 2).

In Fig. 2 such additional notations are accepted: $i = 1...N_r$

$\overline{\alpha} = \{\alpha_i\}$, is estimated raw ore grade; $\overline{\xi} = \{\xi_i\}$ is specific gravity of every variety of ore; $\overline{\rho} = \{\rho_i\}$

$\overline{g} = \{g_i\}$

$j = 1...N_s$ is number of beneficiation stage; N_s – is quantity of stage; $\overline{Q} = \{Q_j\}$, $\overline{C} = \{C_j\}$ is

$\overline{d} = \{d_j\}$ is averaged product coarseness; $\overline{P}_m = \{P_{mj}\}$ is a solid content in pulp; $\overline{B}_m = \{B_{mj}\}, \overline{B}_k = \{B_{kj}\}, \overline{B}_s = \{B_{sj}\}$ are consumption of water to the mill, classifier and magnetic separation respectively; $\overline{\rho}_k = \{\rho_{kj}\}$ is a pulp density in the process of classification; $\overline{\rho}_s = \{\rho_{pj}\}$ is a pulp density before magnetic separation; $\overline{\beta}_{pp} = \{\beta_{pp\,j}\} = \{\beta_j\}$ is an estimated grade in the industrial product; $\overline{\beta}_{\tilde{o}} = \{\beta_{\tilde{o}j}\}$ is loss of a commercial component in tails; β_k is a quality of concentrate; $\overline{\gamma} = \{\gamma_j\}$ is an output of useful component in an industrial product; ©$_k$ is an output of useful component in concentrate; $\overline{\varepsilon} = \{\varepsilon_j\}$ is an extraction of useful component in an industrial product; Σ$_k$ is an extraction of useful component in a concentrate. $\overline{V} = \{v_1, v_2, v_3, ..., v_{n_v}\}$ is a vector of input disturbing parameters (input a priori information); $\overline{U} = \{u_1, u_2, u_3, ..., u_{n_u}\}$ is a control vector (control actions and/or regime parameters); $\overline{Y} = \{y_1, y_2, y_3, ..., y_{n_y}\}$ is a vector of output parameters of the system; n_v, n_u, n_y are corresponding amount of factors.

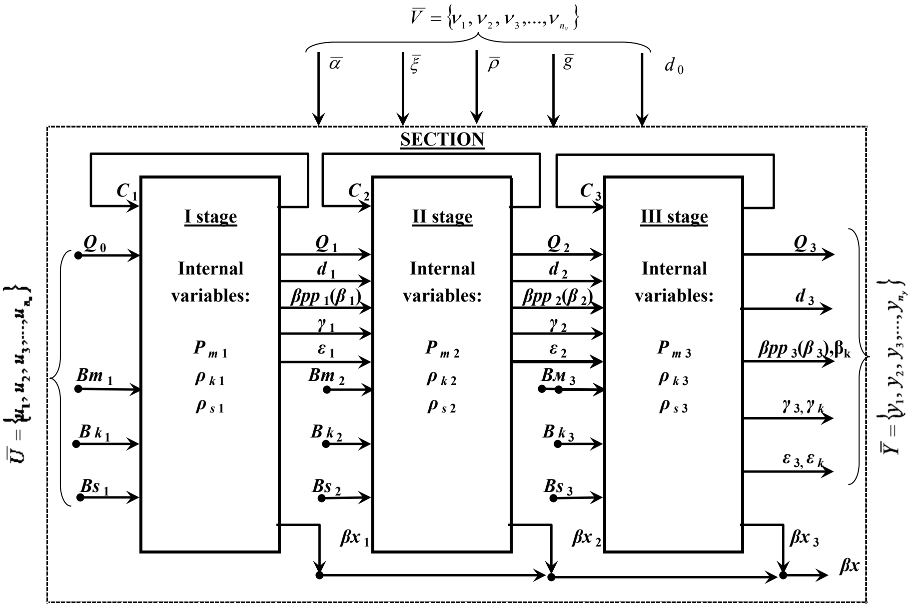


Figure 2: Process line (section) of concentrating as the object of intelligence control [1, 3]

The resulting state vector is

$$\overline{X} = \{\overline{U}, \overline{V}, \overline{Y}\} = \left\{ \begin{array}{l} \overline{V} = \{\overline{\alpha}, \overline{\xi}, \overline{\rho}, \overline{g}, d_0\} \\ \overline{U} = \{Q_0, \overline{Q}, \overline{C}, \overline{d}, \overline{B}_m(\overline{P}_{\hat{o}}), \overline{B}_k(\overline{\rho}_{\hat{e}}), \overline{B}_s(\overline{\rho}_s)\} \\ \overline{Y} = \{\overline{\beta}_{pp}, \overline{\beta}_x, \beta_k, \overline{\gamma}, \gamma_k, \overline{\varepsilon}, \varepsilon_k\} \end{array} \right\}. \quad (1)$$

It should be noted that the indices, such as $\overline{\alpha}, \beta, \gamma, \varepsilon$, can be monitored for several products (e.g., total iron and magnetic properties, etc.). Additionally, the factors $\overline{\alpha}, \overline{\xi}, \rho, \overline{g}, d_0$ (Fig.2) can be considered a priori information, as they are determined in preceding technological processes, such as ore extraction from an open-pit mine or crushing in a crusher, and are not directly controlled (thus, they can be considered as disturbances). Other indices (Fig.2) are generated during the beneficiation process and can be regulated or adjusted based on the specific process conditions. Monitoring of these factors is performed, but not always with the necessary precision and accuracy (particularly for qualitative indicators). Therefore, the distribution of the state vector on input and output indices is conditional, as most of the parameters on output of the first stage will serve as input for the second stage, and so on.

Neural networks are successfully used for the synthesis of such control systems for dynamic objects [3]. Neural networks have a number of properties that make them promising as an analytical tool for control systems. In the context of the task under consideration, this is, first of all, the ability to learn from examples. The presence of large volumes of monitoring data, in which interrelated measurements of inputs and outputs of the system are presented, makes it possible to provide the neural network with representative training samples. Other important properties are the ability of the neural network to adapt to changes in the properties of the control object and the external environment, as well as high resistance to “failures” of individual elements of the network due to the parallelism originally incorporated into its architecture. The ability of a neural network to predict directly follows from its ability to generalize and highlight hidden dependencies between input and output data. Once trained, the network is able to "predict" future output values based on a few previous values and current monitoring data with a high degree of precision.

Within the framework of ongoing research, the use of backpropagation networks seems to be the most promising. Networks of this type generally have significantly shorter training times than backpropagation networks, which allows them to respond more quickly to changes in the conditions of the enrichment process, such as fluctuations in feedstock characteristics, process parameters, or equipment wear. The counter-propagation neural network combines two well-known algorithms: Kohonen's self-organizing map [4] and Grossberg's star [5]. This combination enhances the network's generalizing abilities and enables correct output even with incomplete or slightly distorted input data.

The analysis of the potential of using neural networks for creating models for express analysis of production processes has led to the determination of the neural network model structure. This was achieved by analyzing the technological scheme of the flotation department, taking into account the study's previous results and the conceptual principles of the technology for modeling production processes, such as mineral ore enrichment, which were already adopted within the project.

The parameters used in the model are classified into three groups: benchmarks, control parameters, and indicators. Benchmarks include the characteristics of the input and output products of the flow sheet. The model considers 15 control parameters, which are the parameters that can be influenced to change the conditions for the implementation of the technological process and the values of control indicators. The control parameters in the model include temperature, humidity, pressure, liquid level, voltage, current, etc. Overall, approximately 27 indicative parameters can be considered.

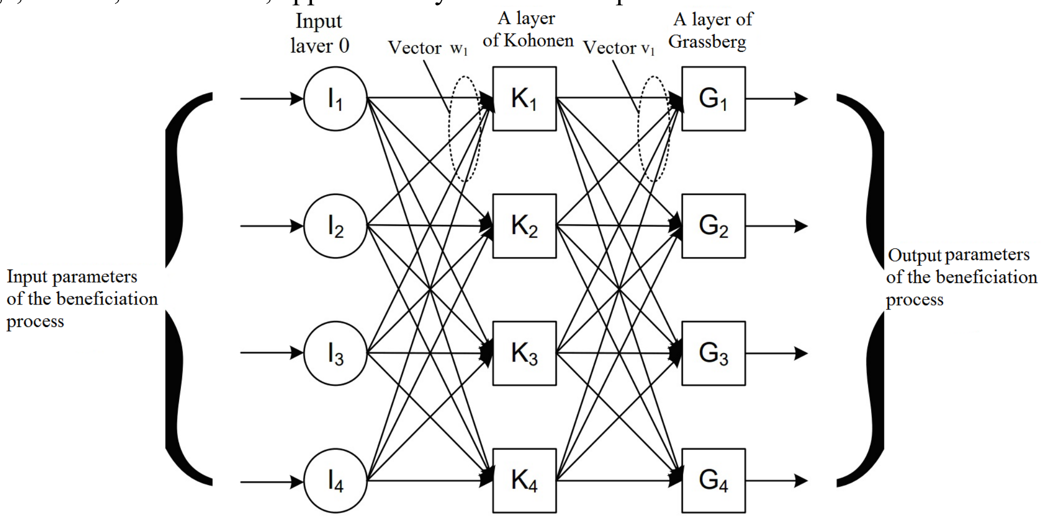


Figure 3: Model of a standard three-layer counter-propagation neural network [7]

Layer 0 neurons do not perform calculations but serve as branching points. Each layer 0 neuron is connected to every neuron in layer 1 (Kohonen's layer), and each neuron in layer 1 is connected to every neuron in layer 2 (Grossberg's fracture). Each connection link has its own weight associated with it. The weights wi of connections of layers 0 and 1 form a matrix of weights W, and the weights VJ of connections of neurons in layers 1 and 2 form a matrix of weights V. The weight values are adjusted in the network training mode, when a priori known vectors of inputs X and outputs Y are fed into the model (Fig. 1). In the predictive mode, the input vector X, which is generated based on the current monitoring data, is fed into the model, and the output vector Y is generated by the network. The output of each layer neuron is simply the sum of the weighted inputs. The Kohonen layer uses a competitive learning process in which the neuron with the highest weighted sum of inputs is selected as the winner. The output of this neuron is assigned the value "1", and the outputs of the remaining neurons of the Kohonen layer are assigned the value "0". The Grossberg layer functions in a similar way - its outputs are determined by the weighted sum of the corresponding inputs from the Kohonen layer. But, since only one neuron of the Kohonen layer has the value “1” set at the output, then in fact each neuron of the Grossberg layer only outputs the value of the weight that connects this neuron with the only non-zero Kohonen neuron. In essence, the Kohonen layer classifies input vectors into similar groups, thereby providing the definition of regions of the multidimensional input space that map to a small neighbourhood of the same “point” in the output space. This is achieved by adjusting the weights of the Kohonen layer, which ensures that the same neuron of this layer is activated by the corresponding input vectors. Before training starts, all weights of the network are assigned some random values. During the learning process, the weight vectors change by "tracking" a small group of input vectors. Training ends when the required pattern of outputs is formed at the output of the neural network. The training of the Grossberg layer is carried out by adjusting only those weights that are associated with a Kohonen neuron that has a non-zero output. The amount of weight correction is proportional to the difference between the weight and the desired output of the Grossberg neuron to which it is connected. The use of a neural network model assumes an a priori classification of the states of the system (beneficiation process) into a finite number of options. Each state in which there is a violation of the procedural characteristics of the process is associated with a set of corrective actions that involve specific changes in control parameters. Both expert assessments and formal classification methods, such as factor and cluster analysis, can be used for classification. The values of output vectors Y are used as the main classification criterion. To determine the current state of the process, a comparison is made between the output of the neural network model and the stored vectors in the system's database that determine the selected states of the enrichment process. If the database indicates that the identified state corresponds to a violation of the regulatory characteristics, then the system retrieves recommendations from the database for correcting the state. If there is an appropriate actuator, the launch of corrective actions can be automated. The developed neural network model of the flotation process was implemented and studied in the Matlab environment [7]. All controlled input parameters are fed to the input of each element of the neural network. The weight coefficients were selected in the process of automatic learning on predetermined samples of real data obtained by the SCADA system as a result of monitoring the production process. In the course of a series of computational experiments, the model was adjusted and provided the synthesis of output vectors corresponding to a control sample of data from a real production process.

3. Research of the effectiveness of methods of the effectiveness of methods

To evaluate the efficiency of radial basis networks and multilayer perceptrons, consider the problem of approximating the function shown in [8]

$$d(x,y) = 3(1-x)^2 \exp(-x^2-(y+1)^2) - 10(\frac{x}{5} - x^3 - y^5 \exp(-x^2-y^2) - \frac{1}{2}\exp(-(x+1)^2 - y^2) , \quad (2)$$

where changing variables within $-3 \le x \le 3$ and $-3 \le y \le 3$. The graph of this function is shown in Fig.3.

Based on a training set of 625 data groups ([x, y], d) generated with a uniform distribution of variables x and y in their domains of definition, a 2-36-1 network structure (2 input neurons, 36 Gaussian-type radial neurons and one output linear neuron). We also used here a hybrid learning algorithm similar to the technique [8]. As a result, the maximum approximation error after 200 iterations was 0.06. Thus, the computational experiment performed showed that the neural network *accurately* restored the function d(x, y) from its tabular values. However, in real conditions, applied problems often arise. For example, they may be associated with the restoration of a function that describes some physical phenomenon. Or such experiments refer to data containing various noise and measurement errors. For this reason, it was decided to repeat the described computational experiment, adapting it to the applied area. This means that the data on which the neural network is built should not be the exact values of the function, as in the mentioned work, but contain some noise. This will make it possible to bring the experiment as close as possible to real problems (e.g., [9-11]).

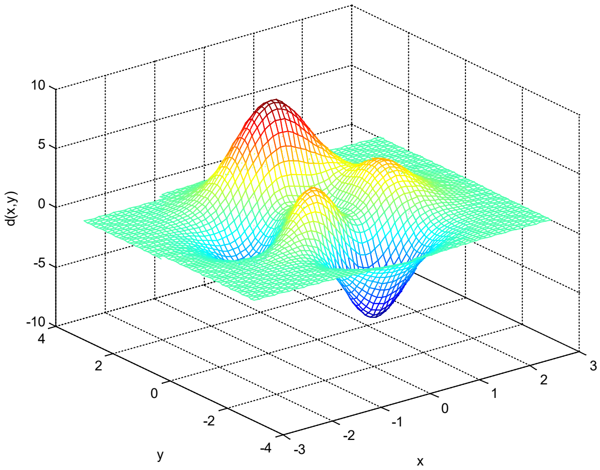


Figure 4: Function Graph d(x,y) [12]

The process of building a neural network model can be divided into 5 main stages (Fig. 5).

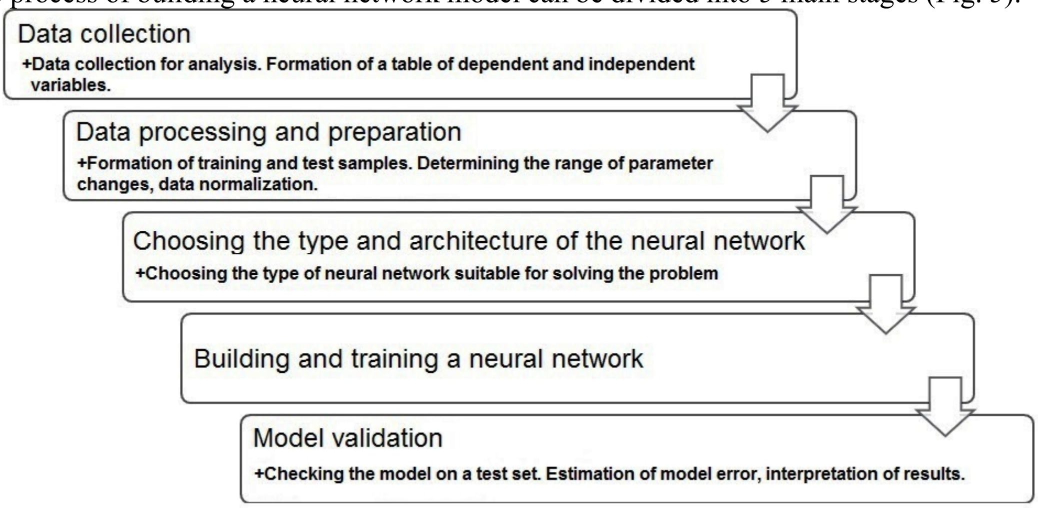


Figure 5: The main stages of the process of building a neural network model [12]

At the first stage, 2 sets of training samples ([x, y], d) containing random noise (errors) were generated in order to simulate the data obtained in the study of a stochastic process or physical phenomenon [12-14]. The first group, d′, contained highly noisy data (noise at the level of 20%), the second, d′′, contained weakly noisy data (noise at the level of 2%). These datasets were generated in the following way. The function d(x,y) is tabulated within $-3 \le x \le 3$ and $-3 \le y \le 3$ with a step of 0.25, and a table of values of the function d is compiled. A certain value ε is added to each value of the function, obtained using a random number generator with a uniform distribution. The expression is used to evaluate

$$-\frac{p\max(|d|)}{2} \le \varepsilon \le \frac{p\max(|d|)}{2}, \tag{3}$$

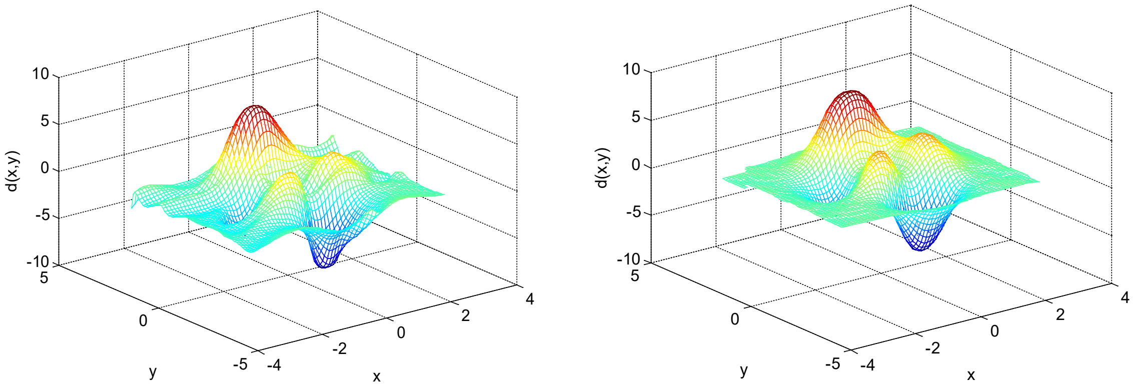
where p is the noise level in fractions of a unit. Thus, the sets of points [x, y, d′] and [x, y, d′′] imitate the results of observations of some physical process containing measurement errors, on the basis of which the process d(x, y) itself will be modelled in a real problem for the researcher unknown.

At the second stage, the data were normalized in the range [-1…1]. In this case, there is no need to divide the total sample into training and testing sets, since the required function d(x,y) is used for testing.

The third stage involves choosing the type of neural network from two suitable for solving the approximation problem: a multilayer perceptron and a radial basic network. In this experiment, both types are used to compare their effectiveness. The architecture of the multilayer perceptron was defined as follows: the network consists of two hidden layers, each containing 8 neurons. The RBF network contains 1 hidden layer, and the number of neurons in this layer grows during the learning process.

At the fourth stage, two specified types of neural networks were trained on each of the data sets. The Neural Toolbox of the Matlab package was used for building and training the neural networks.

At the final stage, response surfaces of the neural network models were built (Fig. 6, 7). The standard deviation, S^2, was used as the error metric for evaluating the resulting models.

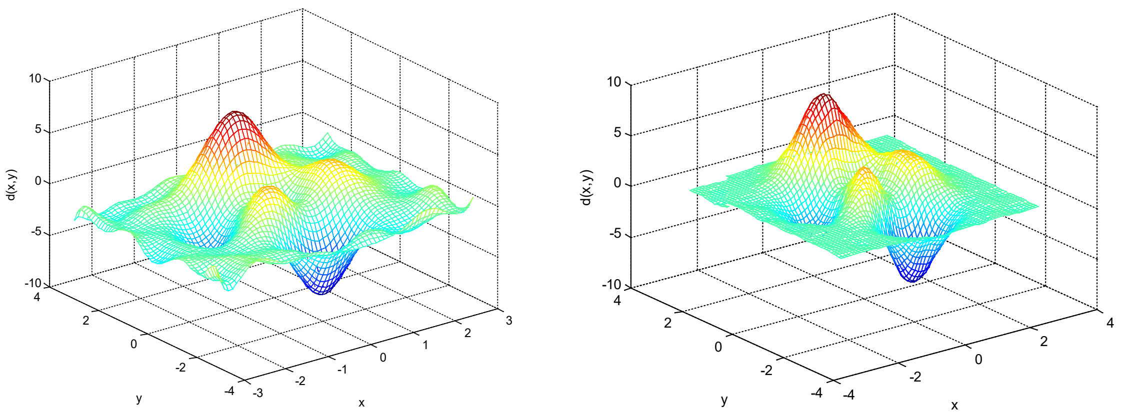


a) b)

Figure 6: The function is approximated by the MLP network: a – based on highly noisy data; b - based on low noise data

In Fig. 6a, the function is modeled on highly noisy data with standard deviation S^2=1,806 (22%) multilayer perceptron. S^2 neural network – 1,63 (20%). The deviation of the neural network from the noisy signal is S^2 =0,844 (9%).

In Fig. 6b function contains weak interference S^2 =0,185 (2%) and approximated by a multilayer perceptron. S^2 neural network – 0,179 (2%). The deviation of the neural network from the noisy signal is S^2 =0,11 (1%).



a) b)

Figure 7: The function is approximated by the RBF network: a - based on highly noisy data; b - based on low noise data

In Fig. 7a function contains noise, S^2 =1,88 (23%) and it is approximated by a radial basis network. S^2 of neural network is 1,68 (20%). Deviation of a neural network from a noisy signal is S^2 =0,85 (10%). On Fig. 7b function contains noise, S^2 =1,185 (2%) and it is approximated by a radial basis network. S^2 of neural network is 0,184 (2%). Deviation of a neural network from a noisy signal is S^2 =0,185 (2%).

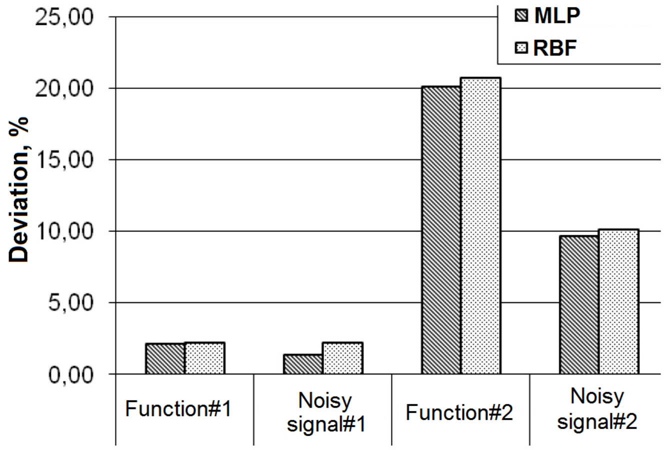
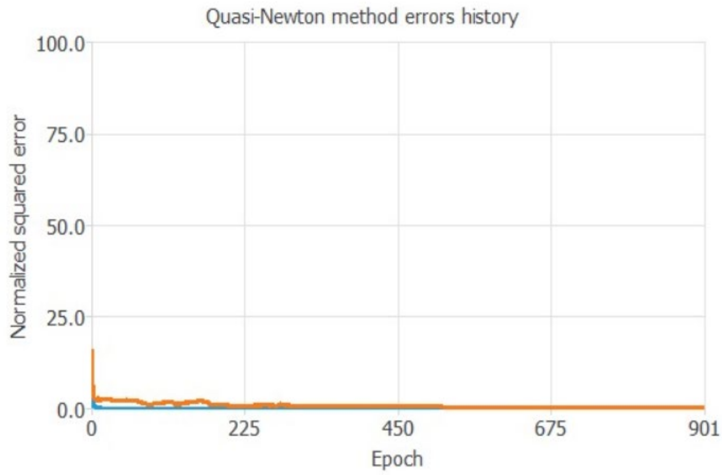


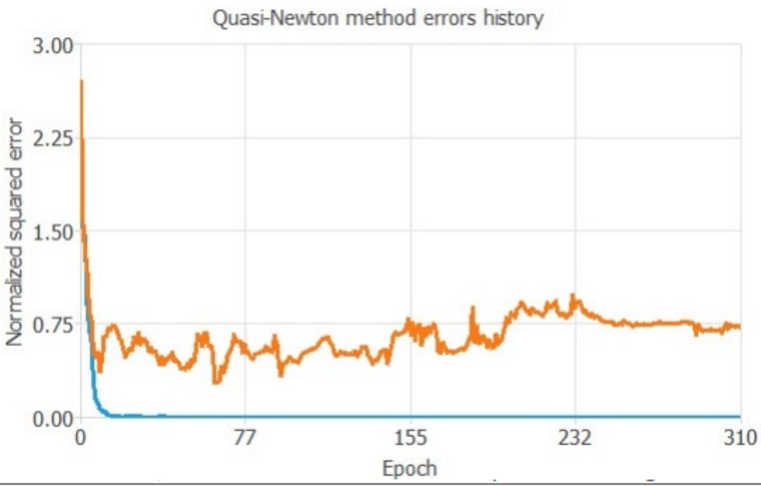
Figure 8: The standard deviation of the considered neural networks

Both types of neural networks considered were able to build a regression model of a noisy signal. In all cases, the networks demonstrated the ability to filter noise similarly to [15-16]. Although the RBF network showed a slightly larger deviation (error) in all cases, the difference from the multilayer perceptron (Fig. 8) is small and insignificant. The radial basis network has an advantage over the multilayer perceptron because it does not require an expert to determine the number of layers and neurons [17, 18]. In the case of a radial basis network, the number of neurons increases during the learning process to achieve a given model accuracy. However, the number of neurons in the RBF network is significantly greater (in this study, by an order of magnitude) than in the perceptron, which slows down working with the RBF network. When predicting the behavior of a function outside the learning range, it is more beneficial to use a multilayer perceptron because it has the ability to extrapolate a function. Numerous studies by the authors [19-22] in various areas of applied neural network technologies confirm similar results.



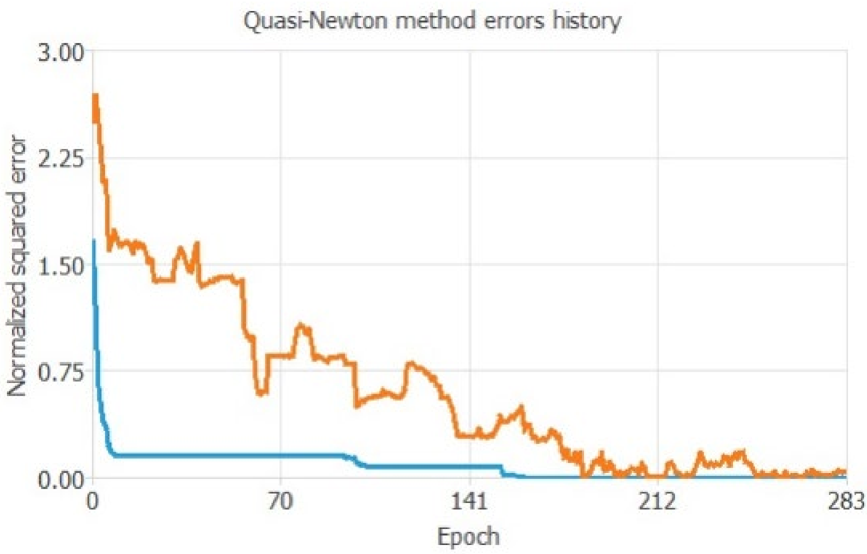
	Value
Final parameters norm	3.61
Final training error	6.18e-06
Final selection error	0.372
Final gradient norm	0.000984

Figure 9: Training schedule for a 3-layer perceptron based on a linear activation function of the output neuron and hidden layers with a hyperbolic activation function (derived by the authors)



	Value
Final parameters norm	12
Final training error	0.000481
Final selection error	0.723
Final gradient norm	0.000997

Figure 10: Training of a 3-layer perceptron based on the logistic activation function of the inner layers (derived by the authors)



	Value
Final parameters norm	5.3
Final training error	0.000159
Final selection error	0.0398
Final gradient norm	0.000959

Figure 11: Training a 3-layer perceptron based on the hyperbolic activation function of all neurons (derived by the authors)

4. Conclusion

Regarding the neural network architectures for approximation and regression analysis, multilayer perceptrons and radial basis networks are both applicable. While each type has its advantages and disadvantages in dependency recovery tasks, both effectively approximate complex functions by learning from noisy data. Multilayer perceptrons have shown good results in processing experimental data, including multidimensional data, making it possible to model patterns hidden within them. In terms of training, a three-layer perceptron based on a linear activation function for the output neuron and hidden layers with a hyperbolic activation function showed the best results in terms of convergence of the learning process and prediction accuracy. Therefore, the aim of this study has been fully achieved.

Future research will be focused on identifying optimal methods for training these neurostructures in real-time. This will include classical gradient algorithms based on error backpropagation (first and second orders [23]) and non-iterative approaches [24]. We also plan to consider alternative neural network architectures, such as recurrent and dynamic structures [25]. These findings will be essential for solving problems related to structural and parametric identification, as well as intelligent control of the separation process (beneficiation or concentration). Based on a priori estimates, this approach has demonstrated sufficient effectiveness in mining plant conditions.

5. References

[1] A. Kupin, Neural identification of technological process of iron ore beneficiation, 2007, 4th IEEE Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS, 2007, pp. 225–227.

[2] A. Bublikov, V. Tkachov, Automation of the control process of the mining machines based on fuzzy logic, Naukovyi Visnyk Natsionalnoho Hirnychoho Universytetu, 2019, 2019(3), pp. 112–118. DOI:10.29202/nvngu/2019-3/19.

[3] A. Kupin, Research of properties of conditionality of task to optimization of processes of concentrating technology is on the basis of application of neural networks. Metallurgical and Mining Industry, 2014, 6(4), pp. 51–55. URL: https://www.metaljournal.com.ua/assets/Journal/11.2014.pdf

[4] V. Morkun, V. Tron, V. Zymohliad, Modelling of Iron Ore Processing in Technological Units Based on the Hybrid Approach Acta Mechanica et Automaticathis, 2022, 16(1), pp. 82–90. URL: https://sciendo.com/it/article/10.2478/ama-2022-0010.

[5] I. Livshin, Artificial Neural Networks with Java, Apress, Berkeley, CA (2019). DOI:10.1007/978-1-4842-4421-0.

[6] F. Vaquero Caballero, D. Ives, Q. Zhuge, M. Sullivan, and S. Savory, Joint Estimation of Linear and Non-linear Signal-to-Noise Ratio based on Neural Networks, Optical Fiber Communications Conference and Exposition, OFC-2018, San Diego, CA, USA, March 11-15, 2018, pp. 1–3. ISBN 978-1-943580-38-5. URL: https://ieeexplore.ieee.org/xpl/conhome/8360560/proceeding.

[7] J.M. Murray, Local online learning in recurrent networks with random feedback, Columbia University, United States, 2019. DOI:10.7554/eLife.43299.

[8] C. Aggarwal Neural Networks and Deep Learning // Radial Basis Function Networks: Springer International Publishing AG, 2018, pp. 217–233. DOI:10.1007/978-3-319-94463-0.

[9] C. Charu, Neural Networks and Deep Learning, IBM Watson Research Center International Business MachinesYorktown HeightsUSA, Springer International Publishing AG, part of Springer Nature (2018). DOI:10.1007/978-3-319-94463-0.

[10] E. Zharikov, S. Telenyk, O. Rolik, Method of Distributed Two-Level Storage System Management in a Data Center Advances in Intelligent Systems and Computing, 938 (2020) 301–315. DOI:10.1007/978-3-030-16621-2_28.

[11] I. Czarnowski, J. Jędrzejowicz, P. Jędrzejowicz, Designing RBFNs Structure Using Similarity-Based and Kernel-Based Fuzzy C-Means Clustering Algorithms, IEEE Access (2021), 9, 4411-4422. DOI:10.1109/ACCESS.2020.3048104.

[12] R. Kruse, S. Mostaghim , C. Borgelt, C. Braune, M. Steinbrecher, Computational Intelligence. A Methodological Introduction, Springer Nature Switzerland AG, 2022, 639 P. DOI:10.1007/978-3-030-42227-1.
[13] K. Du , M. Swamy, Neural Networks and Statistical Learning, Springer London, 2019, 988 P. DOI:10.1007/978-1-4471-7452-3.
[14] R. Silva, J. Menezes, J. Araujo Júnior, Optimization of NARX neural models using particle swarm optimization and genetic algorithms applied to identification of photovoltaic systems, Journal of Solar Energy Engineering, Transactions of the ASME, 143(5): 051001, Oct 2021, p. 051001. DOI:10.1115/1.4049718.
[15] A. Elsheikh, S. Sharshir, M. Abd Elaziz, A. Kabeel, W. Guilan, Z. Haiou, Modeling of solar energy systems using artificial neural network: A comprehensive review, (2019) Solar Energy, 180, pp. 622-639. DOI:10.1016/j.solener.2019.01.037.
[16] F. Itano, M. De Abreu De Sousa, E. Del-Moral-Hernandez, Extending MLP ANN hyper-parameters Optimization by using Genetic Algorithm, (2018) Proceedings of the International Joint Conference on Neural Networks, 2018-July, art. no. 8489520. ISBN: 978-150906014-6, DOI:10.1109/IJCNN.2018.8489.
[17] W. Jiahui, M. Weishi, Z.Zhiyu, Short-Term Load Forecasting Based on GA-PSO Optimized Extreme Learning Machine, (2018) 2nd IEEE Conference on Energy Internet and Energy System Integration, (2018) Proceedings, art. no. 8582545. ISBN: 978-153868549-5, DOI:10.1109/EI2.2018.8582545, URL: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=8564458.
[18] D. Zheng, Y. Pan, K. Guo, H. Yu, Identification and Control of Nonlinear Systems Using Neural Networks: A Singularity-Free Approach, (2019) IEEE transactions on neural networks and learning systems, 30 (9), pp. 2696-2706. DOI:10.1109/TNNLS.2018.2886135.
[19] F. Hutter, L. Kotthoff, J. Vanschoren, Eds., Automated Machine Learning - Methods, Systems, Challenges, (The Springer Series on Challenges in Machine Learning). Basel, Switzerland: Springer, 2019. DOI:10.1007/978–3-030-05318-5.
[20] X. Zheng, MIGO-NAS: Towards fast and generalizable neural architecture search, IEEE Trans. Pattern Anal. Mach. Intell., vol. 43, no. 9, pp. 2936–2952, 2021, DOI: 10.1109/TPAMI.2021.3065138, URL: https://ieeexplore.ieee.org/document/9377468.
[21] N. Pillay, R. Qu, D. Srinivasan, B. Hammer, K. Sorensen, Automated design of machine learning and search algorithms, (2018) IEEE Comp. Intell. Mag., vol. 13, no. 2, pp. DOI:10.1109/MCI.2018.2806988.
[22] Z. Lu, G. Sreekumar, E. Goodman, W. Banzhaf, K. Deb, and V. N. Boddeti, Neural architecture transfer, (2021) IEEE Trans. Pattern Anal. Mach. Intell., vol. 43, no. 9, pp. 2971–2989, DOI:10.1109/TPAMI.2021.3052758.
[23] P. Combettes and J.-C. Pesquet, Lipschitz certificates for layered network structures driven by averaged activation operators, SIAM Journal on Mathematics of Data Science, 2 (2020), pp. 529–557. DOI:10.1137/19M1272780.
[24] S. Han, J. Pool, J. Tran, and W. J. Dally, Learning both weights and connections for efficient neural networks, in Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, NIPS15, Cambridge, MA, USA, 2015, MIT Press, pp. 1135–1143. URL: https://dl.acm.org/doi/10.5555/2969239.2969366.
[25] D. Mocanu, E. Mocanu, P. Stone, P. Nguyen, M. Gibescu, and A. Liotta, Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science, Nature Communications, 9 (2018). DOI:10.1038/s41467-018-04316-3.