Neural network modeling for student competency development*

Yuliia Kostiuk[1,†], Svitlana Rzaieva[1,†], Pavlo Skladannyi[1,2,†], and Volodymyr Sokolov[1,*,†]

[1] *Borys Grinchenko Kyiv Metropolitan University, 18/2 Bulvarno-Kudriavska str., 04053 Kyiv, Ukraine*

[2] *Institute of Mathematical Machines and Systems Problems of the National Academy of Sciences of Ukraine, 42 Ac. Glushkov ave., 03680 Kyiv, Ukraine*

Abstract

The rapid growth of data availability and advances in computing power have positioned neural network modeling and machine learning as essential tools for developing key competencies among higher education students. This paper investigates the application of neurostructural modeling—a generalized extension of traditional feed-forward artificial neural networks—to analyze experimental and observational data, with the primary aim of enhancing students' data analysis, predictive modeling, and decision-making skills in modern educational contexts. A unified theoretical framework is proposed that treats neural network models as compositions of linear and nonlinear structures, incorporating non-classical activation functions (periodic and parameterized forms) and generalized neuron-like elements with flexible connectivity across layers. Special attention is given to constructive (incremental) algorithms for building neurostructural models that guarantee monotonic reduction of the training error. Training is formulated as a nonlinear least-squares problem and addressed through a class of efficient numerical methods based on linear-nonlinear weight decomposition, pseudo-inversion (including block pseudo-rotation via Klines's formula), and Gauss-Newton-like updates with backward-propagation-style Jacobian computation. The developed approach is implemented in a software suite for data storage, extraction, neural model construction, training, and application, including modules for cluster analysis (Kohonen networks), optimal control of dynamic systems with long-term prediction horizons, and analytical processing of large datasets. Experimental validation demonstrates improved computational stability, faster convergence in many cases compared to classical backpropagation, and practical utility for modeling complex systems. The results confirm that integrating neurostructural modeling techniques into higher education curricula significantly strengthens students' competencies in machine learning, data-driven modeling, and adaptive system analysis—skills increasingly demanded in contemporary professional environments. Directions for future work include integration with real-time adaptive learning platforms to support personalized competency development.

Keywords

neural network model, feed-forward neural network, optimal control, higher education student

1. Introduction

The rise of information technologies, particularly in neural network modeling (NNM) and machine learning (ML), has enabled the collection of vast amounts of data [1, 2]. For higher education (HE) students, mastering the analysis of this data is crucial for developing competencies and making informed decisions [3, 4]. Mathematical modeling is essential for uncovering hidden information and for creating reliable models that effectively describe objects and phenomena [5]. NNM automates data analysis and prediction, enabling students to develop skills relevant to a range of fields. However, despite its potential, neural networks face challenges in processing dynamic data. A wide range of neural network tools, from traditional methods to advanced deep learning technologies, demonstrates their applicability in education and beyond. However, ensuring the stability of neural network training remains a challenge, emphasizing the need for robust methods

* *CMIS 2026: 9th International Workshop on Computer Modeling and Intelligent Systems, May 5, 2026, Zaporizhzhia, Ukraine*

* Corresponding author.

† These authors contributed equally.

y.kostiuk@kubg.edu.ua (Y. Kostiuk); s.rzaieva@kubg.edu.ua (S. Rzaieva); p.skladannyi@kubg.edu.ua (P. Skladannyi);v.sokolov@kubg.edu.ua (V. Sokolov)

0000-0001-5423-0985 (Y. Kostiuk); 0000-0002-7589-2045 (S. Rzaieva); 0000-0002-7775-6039 (P. Skladannyi); 0000-0002-9349-7946 (V. Sokolov)

Copyright © 2026 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-07-04

CEUR-WS.org/Vol-4220/paper20.pdf

in data-driven modeling [6]. Rapid growth of data and ML in education; importance of neural modeling for developing student competencies in data analysis, prediction, and decision-making.
Classical backpropagation and deep learning frameworks face challenges in training stability, especially on noisy/small observational datasets common in educational research. Many existing approaches lack a unified theoretical view of linear-nonlinear superposition and constructive (incremental) model building that guarantees monotonic error reduction.

We propose a neurostructural modeling framework that generalizes feed-forward networks using non-classical activations, generalized neuron-like elements with flexible cross-layer connectivity, and constructive algorithms based on linear-nonlinear weight decomposition and block pseudo-inversion (Kline's formula). Training is formulated as a nonlinear least-squares problem solved via efficient Gauss-Newton-like updates with backward-propagation-style Jacobians.

2. Problem Statement

The main objective of this paper is to develop and validate a unified neurostructural modeling framework—a constructive generalization of feed-forward neural networks—that provides improved training stability and convergence compared to classical backpropagation. We demonstrate its application to data-driven modeling tasks relevant to HE and show how integrating these techniques into curricula enhances students' competencies in ML, predictive analytics, and adaptive system analysis. In addition, an additional task is to analyze the effectiveness of NNM and ML methods, based on experimental and observational data, for developing the competencies of HE students. The development of ML in the context of NNM is considered, given the constant accumulation of large amounts of information available via the Internet and the rapid improvement in the efficiency of the computing base. The analysis of changes in the scientific principles of ML focuses on deep learning, which expands the range of ML methods and uses model neurons as basic information-processing units. The importance of data-driven modeling processes in NNM and ML is emphasized [7–9]. Research in this area confirms the great potential of these methods across various fields, including education and science. The prospects of applying NNM and ML to HE students' competencies, based on experimental and observational data, are evident in their ability to adapt and apply across various fields, making them an important tool in the modern educational process.

The scientific problem is a lack of a unified constructive approach to building and training neurostructural models that exploits their linear-nonlinear nature, guarantees monotonic error decrease, and provides better numerical properties than standard backpropagation. The applied/educational problem is how to effectively teach data-driven modeling and ML to higher education students using robust, interpretable tools that work well on real experimental/observational data.

Thus, the following partial buildings can be identified to solve the scientific problem:

1. Develop a theoretical framework treating neural models as compositions of linear and nonlinear structures.
2. Design constructive algorithms and efficient training methods (with mathematical justification).
3. Implement a software suite supporting the full pipeline.
4. Validate computational advantages and applicability to educational modeling tasks (cluster analysis, optimal control, large dataset processing).
5. Analyze implications for student competency development.

3. Methods

Due to the considerable complexity of the objects and phenomena of practical interest, neurostructural modeling methods that develop and generalize neural network methods are

receiving the highest priority [10]. They can be used to solve a wide range of tasks across all fields of activity, including analytical tasks such as predicting situations and managing the development of modeled objects. The class of neural network models includes mathematical models consisting of interconnected basic neuron-like elements and having a characteristic linear-linear superposition structure that depends on the parameters. These include feed-forward neural networks with non-classical activation functions, radial basis functions, probabilistic neural networks, Fahlman neural networks, Takagi-Sugeno fuzzy systems, ANFIS neuro-fuzzy structure models, and others [11]. For such models, the problem of structural identification is partially solved. The use of neural network models and methods does not require prior knowledge of the nature of dependencies, so we can talk about the universality of this mathematical apparatus.

The central stage of building neural network models is training, i.e., setting parameters based on input-output data. To build adequate models, numerical training methods should be used that, unlike most existing methods, take into account the peculiarities of the neurostructural modeling task to the greatest extent possible [12]. The development, implementation, and testing of algorithms for constructing and numerical methods for training models based on computational experimentation using modern computer tools are of both high scientific and practical interest.

To do this, two approaches to assessing effectiveness should be distinguished:

1. Computational with number of iterations to convergence, final training/validation error, numerical stability (e.g., condition number of matrices, avoidance of vanishing gradients).

2. Educational with qualitative/quantitative indicators (e.g., student project outcomes using the software, ability to handle real observational data, comparison with standard tools like TensorFlow/PyTorch in teaching scenarios).

The structure and methodology of applying direct-propagation artificial neural networks were carefully analyzed in the context of their potential for developing competencies in HE students. During the study, the algorithms for training feed-forward neural networks based on numerical methods of local optimization and on methods for solving nonlinear least-squares problems were carefully considered and analyzed. Various approaches to global optimization, algorithms for constructing the optimal structure of feed-forward neural networks, and their possible use for analyzing experimental and observational data have also been studied [13]. It is noted that feed-forward neural networks have a linear-nonlinear structure, determined by the interaction of input data, neuronal activation functions, and weights that reflect the network's internal connections

$$y = y^{(m)} = \sigma^{(m)}\left(\omega_0^{(m)} + W_1^{(m)}\sigma^{(m-1)}\left(\ldots\sigma^{(1)}\left(\omega_0^{(1)} + W_1^{(1)}x\right)\ldots\right)\right) \quad (1)$$

where $x \in R^n$ is inputs; $y \in R$ is output; m is number of layers; $\sigma^{(l)}$ is activation functions of the neurons of the layer; $\omega_0^{(l)} \in R^{N_l}$ is weights of fictitious single inputs; N_l is number of neurons in the l^{th} layer; $\omega_1^{(l)} \in R^{N_l \times N_{l-1}}$ is weights between the neurons of the l^{th} and $(l-1)^{\text{th}}$ layers. The main stage in the construction of feed-forward neural networks is training, the purpose of which is to determine the weights by minimizing the training quality functional

$$J(\omega) = \sum_{i=1}^{k}\left(y_i(\omega) - \tilde{y}_i\right) \quad (2)$$

where k is the number of examples in the training set $\{\tilde{x}_i, \tilde{y}_i\}$, $i = 0, \ldots, k$, $\tilde{x}_i \in R^n$ is the input vector of the i^{th} example, $\tilde{y}_i \in R$ is the teacher's instruction, and $y_i(\omega)$ is the output of the feed-forward artificial neural network for the i^{th} example. Training a feed-forward artificial neural network is a nonlinear least-squares problem involving the weight vector ω and is multi-extreme.

A study of the literature has revealed the existence of other classes of models functionally equivalent to feed-forward artificial neural networks, while at the same time revealing the absence of a single approach to the construction and application of models of this structure. It was found that there are no algorithms for designing and training such a class of models that account for the quadratic nature of the learning-quality functional and the superposition of linear and nonlinear structures with weights [14]. To solve these problems, we formulated the tasks of developing a unified theoretical framework and methods for constructing, training, and applying linear-linear

models similar to artificial feed-forward neural networks, including the development of a set of programs for modeling and analyzing data based on these methods. The concepts of neuron-like elements and neural network models that generalize the concept of artificial neural networks of direct propagation are introduced; the place of neural network models in the structure of mathematical models is considered; a class of nonlinear Voltaire neural networks is introduced; the use of various activation functions is investigated; an algorithm for constructive construction of neural network models is proposed to guarantee monotonicity of reducing the learning error; block recurrent-iterative procedures—algorithms for constructive building and training are developed.

Regarding the application of these methods in HE, their use in the educational process can help create a more effective environment for the development of competencies among HE students. Neural network-based modeling can help analyze and interpret experimental data, enabling students to better understand complex concepts and patterns in their field [15]. ML can help students develop data analysis and processing skills that are important in the modern world. The use of these methods can improve the quality of education and training of qualified specialists across various fields of knowledge.

A neuron is a fundamental component whose functioning is similar to that of an artificial neuron. The neuron converts input $x \in R^n$ to output $y \in R$ with the possible use of weight vectors $\omega \in R^{N_\omega}$ and a priori parameters $\alpha \in R^{N_\alpha}$, where $N_\omega, N_\alpha \geq 0$ by

$$y = \sigma\left(net\left(x, \omega\right), \alpha\right) \tag{3}$$

where $net: R^n \times R^\omega \longrightarrow R$ is neuron activity level—differentiated function by weights ω; $\sigma: R \times R^{N_\alpha} \longrightarrow R$ is differentiated activation function by argument $net(x,\omega)$. In some cases, $net(x,\omega) = \sum_{i=1}^{n} x_i \omega_i$ it is a weighted sum of inputs. In this case, $N_\alpha \succ 0$, the activation function is parameterized. Neural network models are sets of organized neurons connected in layers. Unlike artificial neural networks of direct propagation, the inputs of the neurons of a layer $(i + 1)$ can receive the outputs of neurons $y^{(l,j)}$, $l = 0, \ldots, i$, from any of the previous layers. The activation function is chosen from the set of valid activation functions $\sigma^{(l,j)} \in \Omega$. The set Ω may be limited to a specific class of neural network models used.

The class of neural network models includes models with structures similar to artificial feed-forward neural networks, i.e., neurostructural models (Figure 1).

NNM is a mathematical approach based on neurostructural models, a development and generalization of the artificial neural network approach [16]. Feed-forward artificial neural networks and other types of neural models are widely used to analyze and process experimental and observational data in HE [17].

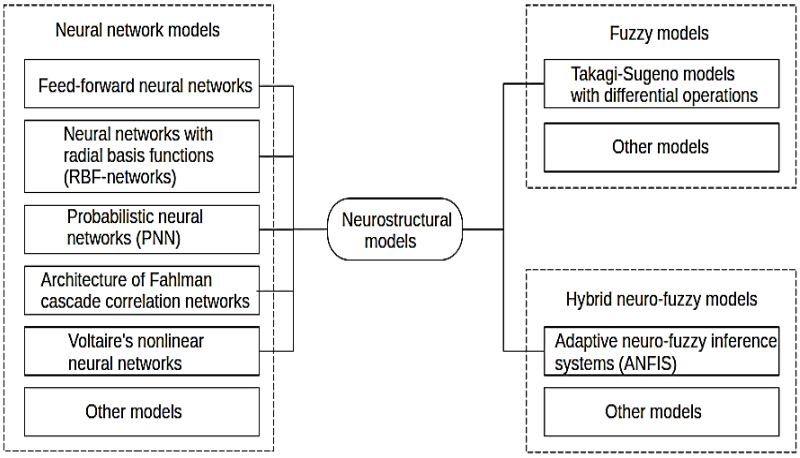

Figure 1: Subclasses of neural network models.

They help HE students develop competencies by modeling complex systems and analyzing data to identify patterns and solve real problems. The use of NNM in the educational process helps to deepen the understanding of the material and the practical application of the acquired knowledge [18].

Models of this type also share a characteristic neurostructural structure. The use of non-classical activation functions allows us to increase the computational capabilities of neurostructural models.

For modeling discrete dynamic systems, it is proposed to use the extension of nonlinear Volterra systems by using nonlinear activation functions

$$y[t]=\omega_0+\sum_{i_l=1}^{L} x[t-i_1]\sigma_{i_1}\left(...\left(\omega_{i_1}\omega_{i_2}+\sum_{i_3=1}^{L} x[t-i_3]\sigma_{i_3}\left(\omega_{i_1 i_2 i_3}+...\right)\right)...\right). \quad (4)$$

The use of periodic trigonometric functions, such as $\sigma(net) = \sin(net)$, in modeling dynamic processes allows us to detect not only the trend but also the seasonal component. Neural networks built on these models are used to analyze and predict the behavior of complex systems from available data, making them a valuable tool for developing HE students' competencies in ML and data analysis [19].

4. Models

For NNM and ML based on experimental and observational data to form the competencies of HE students, it is proposed to use the algorithm shown in Figure 2.

At the initial stage, the type of neural network model is chosen, which determines the specifics of the connections in the neurostructural model and the restrictions on the choice of activation functions. To ensure the monotonicity of model Ω construction, the set must contain a single activation function $\sigma(net) = net$. When expanding the structure to maintain monotonicity and reduce training error, it is necessary to use a special method to set some added weights.

When adding a new neuron to the last hidden layer, its output should be connected to the output neurons with zero weights. When forming a new hidden layer before the original one, its output should be fed back to the original one with unit weights. In the case of fictitious inputs for the original neuronal elements, the corresponding weights should be zero. Block recurrent-iterative procedures are based on the application of Kline's formula for block pseudo-rotation, which is due to the computational advantages of pseudo-rotation of small matrices.

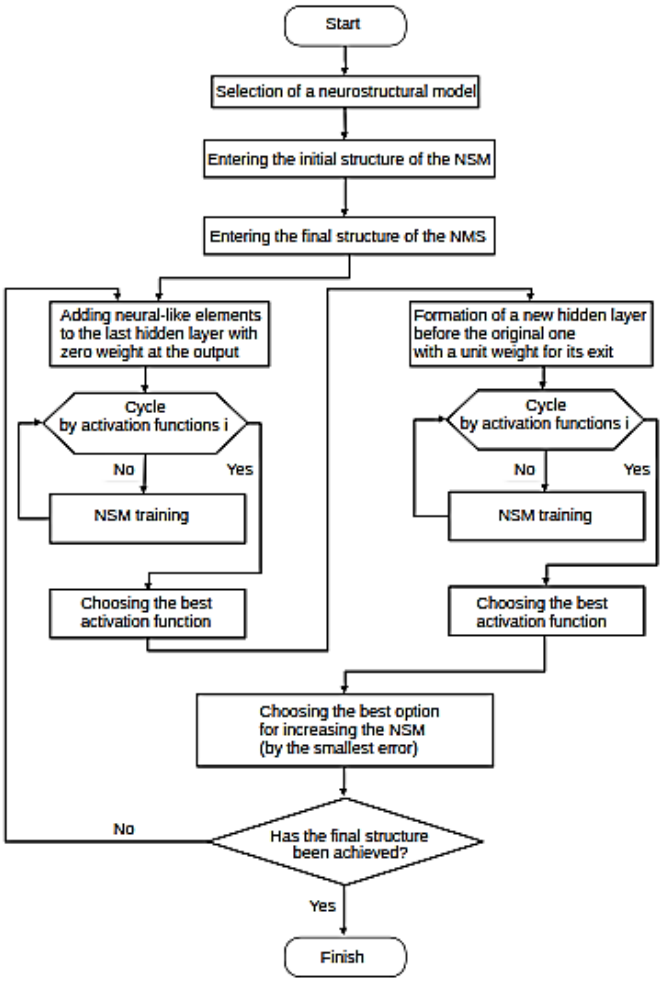

Figure 2: Algorithm for constructing a neurostructural model.

Block-recursive procedures can be used to construct the optimal structure of a neural network model sequentially. When neurons are added to the last hidden layer, an additive addition occurs

$$\hat{y}(\hat{\omega},\omega;x)=\hat{\omega}_{q+1}^{(m)}\sigma\left(\sum_{i=1}^{N_{m-1}}\hat{\omega}_{o,q-1}+\hat{\omega}_{i,q}y^{(m-2,i)}\right) \tag{5}$$

where $\hat{\omega}\in R^{N_{m-2}+2}$ is the vector composed of the weights of the new neuron and the weights from the new neuron to the output neuron; $y^{(m-2,i)}$ is the output of the i^{th} neuron of the $(m-2)^{\text{th}}$ layer. The added function depends on a part of the weight vector ω of the previous model—the weights of the neurons of the hidden layers located from the 1^{st} to the $(m-2)^{\text{th}}$ layer. The new neural network model implements the function

$$y_{\text{new}}(\hat{\omega},\omega;x)=y(\omega;x)+y(\hat{\omega},\omega;x). \tag{6}$$

Adjustment of weights can be expressed in the following form

$$\begin{aligned}\Delta\hat{\omega}&=L(y_{\text{new}}(\hat{\omega}_t,\omega_t)-\hat{y})=L\left(y(\omega_t)+\hat{y}(\hat{\omega}_t)-\hat{y}\right),\\ \Delta\omega&=\left(\nabla_\omega^T y+\nabla_\omega^T\hat{y}\right)+\left(y-\nabla_\omega^T\hat{y}\nabla\hat{\omega}\right)\end{aligned} \tag{7}$$

where $(\cdot)^+$ is the matrix pseudo-inverse operation. Thus, the growth of the vector of the new neural network model depends on the growth of this vector in the previous model and the growth of the neuron's weights

$$\Delta\omega=(\Delta\omega)_y+(\nabla_\omega^T y)^+=\left(\hat{y}-\nabla_\omega^T\hat{y}\nabla\hat{\omega}\right) \tag{8}$$

where $(\Delta\omega)_y$ is the increment obtained for the initial model $y(\omega,x)$, denoted as such to distinguish $\Delta\omega$ it from the new model, the Jacobi matrices $\nabla^T_\omega y$ and $\nabla^T_\omega \hat{y}$ ifor a multilayer neural network model are obtained using an algorithm that accounts for the model's nature, similar to the method of backward error propagation. When a new layer of one neuron is added, a new model is implemented before the original one

$$y_{\text{new}}(\hat{\omega},\omega;x)=\hat{\omega}\,\sigma\big(y(\omega;x)\big). \tag{9}$$

In this case

$$\Delta\omega=\left(\nabla^T_\omega y\right)^+ D^+_k\hat{\omega}^+\left(y-\Psi\Delta\hat{\omega}\right) \tag{10}$$

where $\hat{\omega}^+$ is a scalar ($\hat{\omega}=0$ performed only in some cases), the diagonal matrix $D_{k^*}=diag\{[\sigma'_{y_i}(y_i)]^+\}_{i=1}^k$ is easily determined.

The formula for finding a pseudo-inverse matrix using numerical methods represents a significant contribution to the development of NNM and ML based on experimental and observational data, thereby enhancing the development of competencies among HE students. A class of numerical methods for training a neural network model was developed by decomposing the vector of model weights into linear and nonlinearly entering components using a linear-nonlinear relation based on pseudo-rotation. The software was also developed to test the developed class of numerical learning methods and to compare them with traditional learning methods. The accuracy and efficiency of pseudo-rotation algorithms, including block algorithms, were investigated. The dependence realized by a neural network model with unit activation functions in the neuronal elements of the output layer can be represented as follows

$$y(u,\vartheta;x)=\sum_{j=1}^{q}u_j\psi_j(\vartheta,x)=\psi(\vartheta,x)^T u \tag{11}$$

where $\psi_j(\vartheta,x)$ is the output vector of the i^{th} neuron of the last hidden layer, $\vartheta\in R^p$, $\vartheta\in R^q$ has nonlinear (weights of hidden neural elements) and linear (weights of output neural elements) components of weights, denoted as $\omega = [\vartheta^T,u^T]$. Function (2) can be represented as

$$J(u,\vartheta)=\sum_{i=1}^{k}\left(y(u,\vartheta;\widetilde{x}_i)-\widetilde{y}_i\right)^2=\sum_{i=1}^{k}\left(\sum_{j=1}^{q}u_j\psi_j(\vartheta,\widetilde{x}_i)-\widetilde{y}_i\right)^2. \tag{12}$$

The outputs of the neural network model on the training set are calculated using the formula $y = \Psi(\vartheta)u$, where $\Psi(\vartheta)\in R^{k\times q}$ is the output matrix of the neurons of the last hidden layer. For a fixed vector ϑ, the optimal values of the vector u can be determined using a linear-nonlinear relation

$$u=\Psi(\vartheta)^+\widetilde{y}. \tag{13}$$

The optimization of (2), taking into account the linear-nonlinear relation (13), is equivalent to the problem of weights ϑ

$$\hat{J}(\vartheta)=\|\Psi(\vartheta)\Psi(\vartheta)^+\widetilde{y}-\widetilde{y}\|^2\to\min. \tag{14}$$

To develop a class for training numerical methods of a neural network model, we introduce the notation of a vector

$$H(\vartheta)=\widetilde{y}\left(\Psi(\vartheta)\Psi(\vartheta)^+-1\right)=\Psi(\vartheta)\Psi(\vartheta)^+-I_k\widetilde{y} \tag{15}$$

the Jacobi matrix takes the form

$$H'=\Psi^{+T}(\Psi^T)'\left(I_\rho\otimes\left(I_K-\Psi\Psi^+\right)\widetilde{y}\right)+\left(I_k-\Psi\Psi^+\right)\Psi'\left(I_\rho\otimes\Psi^+\widetilde{y}\right)\in R^{k\times p}. \tag{16}$$

Based on (16), it is possible to implement numerical methods for training a neural network model using methods for optimizing differentiated functions and for solving nonlinear control problems. This approach is used to tune the neural network's parameters to improve its predictive performance. In particular, numerical methods based on the Gauss-Newton algorithm with pseudo-wrapping are used to estimate neural network parameters by minimizing the difference between

predicted and observed values. This allows adjusting the weights and offsets of neurons to improve accuracy and train the model based on experimental data

$$\begin{gathered} K = I_K - \Psi\Psi^+, \\ L = K\Psi'\left(I_\rho \otimes \Psi^+\right) + \Psi^{+\mathrm{T}}\left(\Psi^T\right)'\left(I_\rho \otimes K\right), \\ R' = L\left(I_\rho \otimes \widetilde{y}\right), \\ \Delta\vartheta = -\left[\left(R'\right)^T R'\right]^+ \left(R'\right)^T R \end{gathered} \tag{17}$$

where $\Delta\vartheta$ is the direction of change of the vector ϑ.

The developed class of numerical methods is extended to multi-output neural network models by representing the resulting connection matrix using $H(\vartheta) = \Psi(\vartheta)\Psi(\vartheta)^+\tilde{Y} - \tilde{Y}$, the vectorization operation "vec." The functional is to be minimized

$$\left\|\operatorname{vec} H\left(\vartheta\right)\right\|^2 = \left\|\operatorname{vec}\left(\Psi\left(\vartheta\right)\Psi\left(\vartheta\right)^+ \widetilde{y}\right) - \operatorname{vec}\hat{Y}\right\|^2. \tag{18}$$

The developed class of numerical methods can be applied to the training of multilayer neural networks. To find the Jacobi matrix $\Psi(\vartheta)$, we propose an algorithm that takes into account the positional nature of neural networks. We have a derivative

$$\frac{\partial y_i^{(m-1)}}{\partial \omega_j^{(h,k)}}, i = 1, \ldots, N_{m-1}, h = 1, \ldots., m-2, k = 1, \ldots., N_h \tag{19}$$

where $y_i^{(m-1)}(\vartheta, x)$ is the output of the i^{th} neuron of the last hidden, $(m-1)^{\text{th}}$ layer; weight $\omega_j^{(h,k)}$ is the weight of the k^{th} neuron of the h^{th} layer, defined as follows

$$\begin{gathered} \frac{\partial y_i^{(m-1)}}{\partial \omega_j^{(h,k)}} = \frac{\partial y_i^{(m-1)}}{\partial y_k^h} \frac{\partial y_k^h}{\partial net_j^{(h,k)}} \frac{\partial net_j^{(h,k)}}{\partial \omega_j^{(h,k)}}, \\ s^{(i,h,k)} = \frac{\partial y_i^{(m-1)}}{\partial y_k^h} = \sum_{l=1}^{N_{h+1}} s^{(i,h+1,l)} \frac{\partial y_l^{(h+1)}}{\partial y_k^h}, h = m-2, \ldots, 1. \end{gathered} \tag{20}$$

Determining the optimal solution using the developed class of numerical methods for training a neural network model typically takes longer. However, the number of iterations required to solve the problem is lower.

The basis of the methods for training a neural network model based on decomposition is the pseudo-inverse operation. To increase the efficiency of computational procedures, the following approaches are possible: using the most stable and efficient algorithm for pseudo-inverse matrices, and using the block pseudo-inverse matrix operation based on the Kline formula.

The developed universal approach to data analysis using information from information system databases is a key element in NNM and ML. This approach includes developing a universal data warehouse structure and a set of programs for an information-analytical system [20]. The software is used to administer a universal data warehouse and to process operational and analytical data. For effective control of dynamic systems, NNM methods that account for the model's structure are used. The proposed control algorithm is based on these methods. HE institutions can use these tools to develop the competencies of HE students by leveraging access to a universal data warehouse and the transactional databases of the information system. This allows for practical data analysis and preparation of HE students for the requirements of the modern labor market. Figure 3 shows the structure of the complex of programs for data analysis.

The management system is designed to efficiently manage users, metadata, specifications, available areas, and the data upload process to the repository. To analyze the data, an information-analytical system has been created that enables data extraction from the repository for further analysis, including extraction, display, and detailing [21]. The proposed analysis methodology is based on the consistent detailing of information extracted from databases using SQL queries, presented in two-dimensional tables.

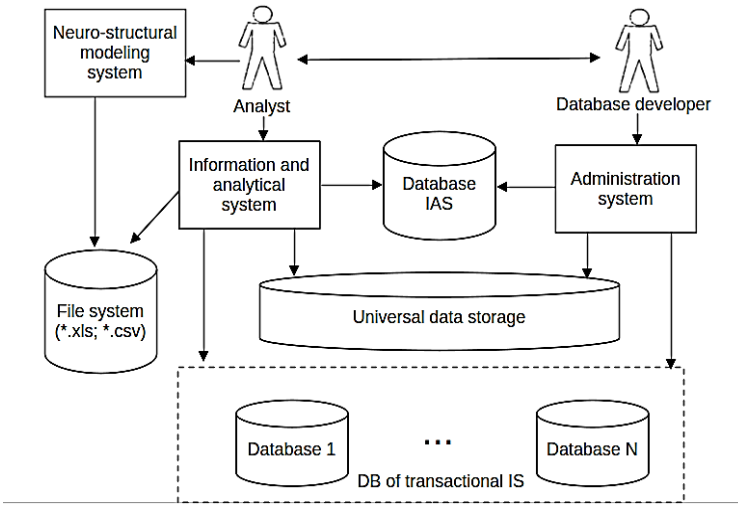

Figure 3: Structure of the information and analytical system.

The software for NNM and data analysis is designed to perform analytical calculations and evaluations using neural network models. To expand this software's functionality, several algorithms were developed, including one to automatically determine the optimal number of clusters to reduce the size of the training set by applying the Kohonen neural network [22]. This information helps understand how systems based on NNM and ML can be used to develop the competencies of HE students, allowing them to work with real data and solve complex analytical and predictive problems [23].

Another challenging and important task of using NNM in analytical data processing is the synthesis of optimal control of dynamic systems whose model has the form

$$y[t]=f(\omega; x[t], x[t-1], \ldots, x[t-d]; y[t], y[t-1], \ldots, y[t-d]) \tag{21}$$

where d is the order of signal delay, let $u[t]$ be the vector of control influences, which is a subvector of $x[t]$. Let's consider the functionality

$$J(u[T+1], \ldots, u[T+S]) = \sum_{t=T+1}^{T+S} g(u[t], y[t]) \tag{22}$$

where T is the last moment at which the output value is known; S is the control bias period; g is a function chosen depending on the task, differentiated by the elements of the vectors $y[t], u[t], t = T + 1, \ldots, T + S$. We consider maximization by the values of $u[T + 1], \ldots, u[T + S]$. This $S \succ 1$ raises the problem of optimal control with a bias, i.e., control that accounts for the long-term influence of control actions $u[t]$ on the system's behavior (21). The developed algorithm for optimal control based on NNM accounts for the nature of neural network models of control objects. Partial derivatives for (22)

$$\frac{\partial J}{\partial u_i[s]} = \sum_{t=s}^{T+S} \frac{\partial g(u[t], y[t])}{\partial u_i[s]}, \tag{23}$$

$$\frac{\partial g(u[t], y[t])}{\partial u_i[s]} = \sum_{r=1}^{R} \frac{\partial g(u[t], y[t])}{\partial u_r[t]} \cdot \frac{\partial y_r[t]}{\partial u_i[s]}. \tag{24}$$

The factor $\frac{\partial g(u[t], y[t])}{\partial u_r[t]}$ in (24) is determined from the form of the function g. For $t = s$ an arbitrary layer $m = 0, \ldots, M - 1$, we obtain

$$\frac{\partial y_r}{\partial y^{(m,i)}} = s_{mi}^r = \sum_{q=1}^{N_{m+1}} \frac{\partial y_r}{\partial y^{(m+1,q)}} \cdot \frac{\partial y^{(m+1,q)}}{\partial y^{(m,i)}} = \sum_{q=1}^{N_{m+1}} s_{m+1,q} \cdot \frac{\partial y^{(m+1,q)}}{\partial y^{(m,i)}}. \tag{25}$$

Formula (25) allows us to recursively calculate the value of S_{mi}^{r}, starting with the layer $m = M - 1$ and decreasing m. The derivative $\frac{\partial y_r[t]}{\partial u_i[s]} = \frac{\partial y_r[t]}{\partial y^{(0,r)}} = s_{0i}^{r}$ is obtained at $m = 0$. In this case

$$\frac{\partial y^{(m+1,q)}}{\partial y^{(m,i)}} = \frac{\partial y^{(m+1,q)}}{\partial net^{(m+1,q)}} \frac{\partial net^{(m+1,q)}}{\partial y^{(m,i)}} = \sigma'_{net}\left(net^{(m+1,q)}\right) \cdot \omega_i^{(m+1,q)} \tag{26}$$

where $net^{(m+1,q)}$ is the derivative of the activation function by its argument, which represents the level of activity of the neurostructural element, the initial condition is also calculated according to formula (26). We $t \succ s$ have

$$\frac{\partial y_r[t]}{\partial u_i[s]} = \frac{\partial y_r[t]}{\partial u_i[s]_1} + \sum_{p=1}^{R} \frac{\partial y_r[t]}{\partial y_p[s]} \cdot \frac{\partial y_p[s]}{\partial u_i[s]} \tag{27}$$

where $\frac{\partial y_p[s]}{\partial u_i[s]}$ and $\frac{\partial y_r[t]}{\partial u_i[s]_1}$ are the derivatives of the function (21), which depend on $u_i[s]$, they are calculated by formulas (25) and (26), as well as in (27) $\frac{\partial y_r[t]}{\partial y_p[s]}$.

Figure 4 shows the functional structure of the software for neurostructural modeling. The structure includes various modules and components that provide data analysis, neural network model construction, and training, taking into account their linear and nonlinear nature [6]. Each module performs a specific function in the development and application of neural network models, contributing to their effective use in practical research and the solution of complex analytical problems [5].

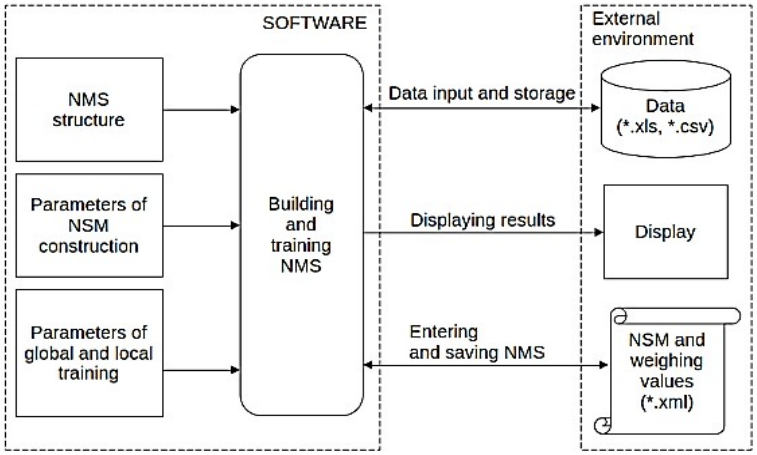

Figure 4: Software functional structure.

Thus, a neurostructural approach to modeling complex systems, based on a unified view of their linear and nonlinear structures and accounting for weights during model construction, training, and application, is investigated. This methodology uses neural networks as a key tool for analyzing and modeling complex systems. In addition, practical algorithms for solving modeling problems have been developed, and a software suite for neurostructural data analysis has been created. The use of these approaches can help develop competencies among HE students in NNM and ML, expanding their knowledge and skills in these areas.

5. Conclusions and Future Work

The study developed and tested a numerical method for training neural network models, improving the algorithm for optimal control of neural structures in solving analytical problems. A set of programs for data modeling and analysis using these methods was also developed, enabling HE students to develop competencies in NNM.

The research demonstrates that ML and neural network methods can significantly enhance competency formation by providing more efficient models. Optimized algorithms improve computational efficiency during model training, addressing the needs of modern education and supporting decision-making with large datasets.

NNM and ML are crucial in modern science and education, offering high efficiency in skill development. The research confirms the significant potential of these methods in solving complex problems across various fields, including education, and highlights their adaptability as key tools in HE.

Further research could explore integrating NNM with real-time adaptive learning systems to enhance personalized education and competency development in HE.

Declaration on Generative AI

While preparing this work, the authors used the AI programs Grammarly Pro to correct text grammar and Strike Plagiarism to search for possible plagiarism. After using this tool, the authors reviewed and edited the content as needed and took full responsibility for the publication's content.

References

1. V. Buriachok, et al., Implementation of Active Cybersecurity Education in Ukrainian Higher School, in: Lecture Notes on Data Eng. and Commun. Technol., 2023, 533–551. doi:10.1007/978-3-031-35467-0_32

2. V. Buriachok, V. Sokolov, Implementation of Active Learning in the Master's Program on Cybersecurity, in: Advances in Intell. Syst. and Comput. Springer Int. Publishing, 2019, 610–624. doi:10.1007/978-3-030-16621-2_57

3. B. Bebeshko, et al., Application of Game Theory, Fuzzy Logic and Neural Networks for Assessing Risks and Forecasting Rates of Digital Currency, J. Theor. Appl. Inf. Technol. 100(24), 2022, 7390–7404.

4. K. Khorolska, et al., Application of a Convolutional Neural Network with a Module of Elementary Graphic Primitive Classifiers in the Problems of Recognition of Drawing Documentation and Transformation of 2D to 3D Models, J. Theor. Appl. Inf. Technol. 100(24), 2022, 7426–7437.

5. G. Sun, L. Zhao, The Construction of Competency Training Mechanism Model for Tourism Undergraduates based on Grounded Theory, PLOS ONE 19(2), 2024, e0296683. doi:10.1371/journal.pone.0296683

6. A. Zinilli, G. Cerulli, Link Prediction and Feature Relevance in Knowledge Networks: A Machine Learning Approach, PLOS ONE 18(11), 2023, e0290018. doi:10.1371/journal.pone.0290018

7. L. J. Anastasopoulos, A. B. Whitford, Machine Learning for Public Administration Research, with Application to Organizational Reputation, J. Pub. Adm. Res. Theory 29(3), 2018, 491–510. doi:10.1093/jopart/muy060

8. J. E. Sierra-García, M. Santos, Switched Learning Adaptive Neuro-Control Strategy, Neurocomp. 452, 2021, 450–464. doi:10.1016/j.neucom.2019.12.139

9. G. Montavon, W. Samek, K.-R. Müller, Methods for Interpreting and Understanding Deep Neural Networks, Digital Signal Process. 73, 2018, 1–15. doi:10.1016/j.dsp.2017.10.011

10. S. Amri, H. Ltifi, M. Ben Ayed, A Predictive Visual Analytics Evaluation Approach based on Adaptive Neuro-Fuzzy Inference System, Comput. J. 62(7), 2018, 977–1000. doi:10.1093/comjnl/bxy091
11. H. Ltifi, S. Amri, M. Ben Ayed, Fuzzy Logic-based Evaluation of Visualizations Generated by Intelligent Decision Support Systems, Inf. Visualization 17(1), 2016, 3–21. doi:10.1177/1473871616674046
12. T. Wang, Y. Yang, W. Xiang, Computationally Efficient Neural Hybrid Automaton Framework for Learning Complex Dynamics, Neurocomp. 562, 2023, 126879. doi:10.1016/j.neucom.2023.126879
13. Z. Zhu, et al., Integrating Reinforcement Learning with Deterministic Learning for Fault Diagnosis of Nonlinear Systems, Neurocomp., 562, 2023, 126847. doi:10.1016/j.neucom.2023.126847
14. Y. Kostiuk, et al., Research of Methods of Control and Management of the Quality of Butter on the basis of the Neural Network, in: 2022 Int. Conf. on Smart Inf. Syst. and Technol. (SIST), 2022. doi:10.1109/sist54437.2022.9945764
15. Y. Yang, W. Xiang, Modeling Dynamical Systems with Neural Hybrid System Framework via Maximum Entropy Approach, in: 2023 IEEE Am. Control Conf., 2023. doi:10.23919/acc55779.2023.10155820
16. W. Xiang, Runtime Safety Monitoring of Neural-Network-Enabled Dynamical Systems, IEEE Trans. Cybernet. 52(9), 2022, 9587–9596. doi:10.1109/tcyb.2021.3053575
17. Y. Smitiukh, et al., Development of a Prototype of an Intelligent System for Predicting the Quality of Dairy Manufacture, in: 2022 IEEE 11th Int. Conf. on Intell. Syst. IEEE, Oct. 12, 2022. doi:10.1109/is57118.2022.10019699
18. W. Xiang, H.-D. Tran, X. Yang, and T. T. Johnson, "Reachable Set Estimation for Neural Network Control Systems: A Simulation-Guided Approach," IEEE Trans. Neural Netw. Learn. Syst., vol. 32, no. 5. IEEE, pp. 1821–1830, May 2021. doi:10.1109/tnnls.2020.2991090
19. H. Dou, F. Shen, J. Zhao, and X. Mu, "Understanding Neural Network through Neuron Level Visualization," Neural Netw., vol. 168. Elsevier, pp. 484–495, Nov. 2023. doi:10.1016/j.neunet.2023.09.030
20. E. Tjoa, H. J. Khok, T. Chouhan, C. Guan, Enhancing the Confidence of Deep Learning Classifiers via Interpretable Saliency Maps, Neurocomp. 562. 2023, 126825. doi:10.1016/j.neucom.2023.126825
21. J. Bilski, et al., Fast Computational Approach to the Levenberg-Marquardt Algorithm for Training Feedforward Neural Networks, J. Artif. Intell. Soft Comp. Res. 13(2), 2023, 45–61. doi:10.2478/jaiscr-2023-0006
22. J. Heer, A. Perer, Orion: A System for Modeling, Transformation and Visualization of Multidimensional Heterogeneous Networks, Inf. Visualization 13(2), 2012, 111–133. doi:10.1177/1473871612462152
23. S. Kiranyaz, et al., Self-organized Operational Neural Networks with Generative Neurons, Neural Netw. 140, 2021, 294–308. doi:10.1016/j.neunet.2021.02.028