DOI: https://doi.org/10.15407/kvt206.04.005

UDC 0004.021

MOROZ O.H., PhD (Engineering),
Senior Researcher of the Dept. for Information
Technologies of Inductive Modeling
ORCID: 0000-0002-0356-8780,
e-mail: olhahryhmoroz@gmail.com
STEPASHKO V.S., DSc (Engineering), Professor,
Head of the Dept. for Information
Technologies of Inductive Modeling
ORCID: 0000-0001-7882-3208,
e-mail: stepashko@irtc.org.ua
International Research and Training Centre
for Information Technologies and Systems
of the National Academy of Sciences of Ukraine
and Ministry of Education and Science of Ukraine,
40, Acad. Glushkov av., Kyiv, 03187, Ukraine

COMPARATIVE FEATURES OF MULTILAYERED ITERATIVE ALGORITHM GMDH AND DEEP FEED-FORWARD NEWURAL NETWORKS

Introduction. *Deep neural networks are effective tools for solving actual tasks such as data mining, modeling, forecasting, pattern recognition, clustering, classification etc. They differ with respect to the architecture design, learning methods and so on. Most simple and widely used are deep feed-forward supervised NNs.*

The purpose of the paper *is to compare briefly main features of the deep feed-forward deterministic supervised networks with the Multilayered Iterative Algorithm of GMDH (MIA GMDH) and to formulate main ideas of constructing a new class of hybrid deep networks based on the MIA neural network.*

Methods. *Most usable deep feed-forward supervised neural networks have been studied: multilayered perceptron, convolutional NN and some its modifications, polynomial neural networks, genetic polynomial neural network etc.*

Results. *There was carried out a comparative analysis of main features of the MIA GMDH neural network with the characteristics of other deep deterministic supervised neural networks. The most promising approaches are identified to improve the performance of this network, particularly by hybridization with methods of computational intelligence. The main idea of building a new class of hybrid deep networks based on MIA GMDH is formulated.*

Conclusions. *MIA GMDH and its modifications are original representatives of the self-organizing networks potentially giving best results, especially for big data case. Hybridization of GMDH-based NNs with stochastic methods of computational intelligence is suggested to achieve a synergetic effect.*

Keywords: *multilayered iterative algorithm of GMDH (MIA GMDH), self-organizing neural network, neural network architecture, deep neural networks, feed-forward neural networks, supervised neural networks, deep learning.*

INTRODUCTION

Deep neural networks (DNNs) [1] are effective tools for solving actual tasks such as forecasting, data mining, pattern recognition, clustering, classification etc. Under the term "deep neural networks" is most often understood a neural network with more than two hidden layers [2].

There are many various deep neural networks most usable of which are presented in [1, 3–7]. As to the learning methods, they may be divided into:

— supervised (algorithms are learned to predict outputs from the input data and all observations of a dataset are labeled);

— unsupervised (the internal structure is determined from the input data and all the observations belonging in a dataset are unlabeled);

— semi-supervised (only part of the dataset observations are labeled);

— with reinforcement learning (labeled input/output pairs are not needed, the machine is allowed to interact with its environment and is "rewarded" when it performs the task correctly).

By architecture design neural networks (NN) generally are divided into feed-forward and feedback neural networks (with memory cells). In this paper, the emphasis is placed only on deep feed-forward supervised NNs.

ANALYSIS OF THE PROBLEM

Deep feed-forward supervised NNs include *deterministic* NNs such as multilayered perceptron, extreme learning machines, liquid state machines, convolutional NN and some its modifications (Deconvolutional NN, Π-nets, residual network, GoogleNet, AlexNet, VGGNet). Besides, there are used networks with *probability rules* such as capsule network, deep belief networks, deep convolutional inverse graphics networks etc.

At the same time, there are known algorithms realizing principles of the Group Method of Data Handling (GMDH) and being typically interpreted as the Polynomial Neural Networks of deterministic type and the GMDH-based Genetic polynomial neural networks of probabilistic type.

Generally, modern deterministic deep feed-forward networks have such main disadvantages as overfitting, slow operation speed, a large amount of the needed computing resources. There is a need to search for such architecture of a neural network and methods for its training which could help to eliminate these shortcomings without appearing new ones.

PROBLEM STATEMENT

It is important to study the features of the deep networks, find ideas that contribute to the elimination of some of the above shortcomings, which could help to create more efficient neural networks. Only strongly deep deterministic feed-forward neural networks (with more than three hidden layers) are considered below and compared with the architecture of the Multilayered Iterative Algorithm of GMDH (MIA GMDH) to determine most promising approaches to improving the deep NNs using some GMDH ideas.

Hence, the **purpose of this paper** is to compare briefly main features of the deep feed-forward deterministic supervised networks with the MIA GMDH neural network and to formulate main ideas of constructing a new class of hybrid deep networks based on MIA GMDH.

CHARACTERISTICS OF THE DEEP FEED-FORWARD DETERMINISTIC NEURAL NETWORKS

Multilayered perceptron (MLP) [9–11] feeds information from the input layer of the network to its output layer, and there is no feedback between the layers until inputs arrives at the outputs. The MLP consists of a large number of perceptrons organized in layers and each unit in a layer connects to all other neurons from the previous layer. Each connection is a parameter of the network. These connections are not all equal and can differ in strengths or weights. The weights of these connections represent the knowledge of the network. The MLP consist of three kinds of layers: 1) input layer with raw data; 2) hidden layer(s) with sequences of functions to be applied to the previous hidden layers inputs or outputs; 3) the output layer is a finite function or a set of functions.

The MLP can use linear or non-linear activation functions. Every layer node is a fully connected one, and in some realizations the nodes number in every layer is the same. In many realizations the activation function across hidden layers is identic. To find optimal parameters, a learning rule is needed. A general approach is to determine an error function along with an optimization algorithm for finding optimal parameters by minimizing the error for training data.

The MLP like most others feed-forward NN is trained by a back-propagation algorithm, although the Stochastic Gradient Descent is presently used for computational efficiency. The error being back-propagated is often some change of the difference between inputs and the outputs. Given that the NN has sufficiently hidden neurons, it can theoretically model ever relationship between the input and output. Practically, their use is a lot more limited but they are popularly hybridized with other networks for forming new NNs.

The Stochastic Gradient Descent computes a gradient for a set of randomly chosen training samples (batch) and updates the parameters for this batch sequentially. This leads to a quicker learning, a drawback is a growth in inaccuracy. But for big data sets the speed edge outweighs this drawback. Using the gradient descent leads to the overfitting that occurs when a function corresponds too closely to a particular set of data and the model cannot give correct results on samples that were not used as training samples.

The MLP output is obtained from the input as a model which can be usable without knowledge how the output should be structured to build a relatively fast and easy NN. The MLP NNs are good for both classification and prediction.

Convolutional NN (ConvNet) [12–15] receives images as an input, use them for training a classifier and consist of eight weight layers [13]. In this CNN a neuron is only connected to nearby neurons in the next layer to essentially reduce the full parameters number in the network.

Convolutional Layer applies a convolution filter to the image for detection image features. In convolutional layers each node only concerns itself with close neighboring cells. These convolutional layers also tend to shrinking with an increase in the number of layers, by easily divisible input factors.

Each convolutional layer contains a series of filters known as convolutional kernels. The filter is a matrix of integers that are used on a subset of the input pixel values of the same size as the kernel. Each pixel is multiplied by corresponding value in the kernel, and then the result is summed up for a single value for simplicity representing a grid cell, like a pixel, in the output channel/feature map. These are linear transformations and each convolution is a type of affine function. In computer vision the input is often a three channel RGB image. The kernel strides over the input matrix of numbers moving horizontally column by column, sliding/scanning over the first rows in the matrix containing the images pixel values. Then the kernel strides down vertically to subsequent rows. Note, the filter may stride over one or several pixels at a time. In other non-vision applications, a one-dimensional convolution may slide vertically over an input matrix.

In CNN models there are often more than three convolutional kernels, 16 kernels or even 64 kernels in a convolutional layer are common. These different convolution kernels act as a different filter creating a channel/feature map representing something different. For example, kernels could filter top edges, bottom edges, and diagonal lines and so on. In much deeper networks these kernels could be filtered to animal features such as eyes or bird wings. Having a higher number of convolutional kernels creates a higher number of channels/feature maps and a growing number of data and uses more memory due to this.

A kernel will be shared totally with other neurons that are connected to their local receptive fields. The results of these calculations between the local receptive fields and neurons using the same kernel will be saved in a matrix defined as an activation map. Various kernels will result in various activation maps, and the kernels number can be regulated with hyper-parameters.

In this way, regardless of the full number of connections between neurons in a network, the full number of weights corresponds only to the size of the local receptive field (the size of the kernel). The convolutional network architecture generally consists of four types of layers: convolution, pooling, activation and fully connected.

ReLU Activation Layer. The convolution maps are passed through a nonlinear activation layer (such as Rectified Linear Unit (ReLu), tanh, sigmoid etc.) which substitutes filtered images negative numbers with zeros and often applied to the values from the convolutional operations between the input and the kernel. These values are stored in the activation maps, which will be passed to the next network layer. The shared local connectivity and weights help essentially in reducing the full network parameters number.

Besides, shared local connectivity and weights are important in effective images processing. Using a kernel that shares identic weights can find patterns from all the local regions in the image and various kernels can recover various kinds of patterns of the image.

Pooling Layer aims at decreasing the dimension of the input vector with some pre-specified pooling method with conserving as much information as possible. Also a pooling layer is able to introduce spatial invariance into the network to improve the model generalization. To carry out pooling, such layer uses zero-padding, stride and a size of pooling window as hyper-parameters.

Pooling layers help to monitor overfitting by reducing the number of parameters and calculations in the network. There are many kinds of pooling methods, such as min-pooling, averaging-pooling, stochastic pooling and fractional max-pooling. The most usable pooling method is max-pooling which to be superior in dealing with images by capturing invariances efficiently. Max-pooling recovers the maximum value within each specified sub-window across the activation map.

Fully Connected Layers. After a few iterations of pooling and convolution layers, before output layer there is a classical MLP for further processing the data and to further model non-linear relationships of the input features. These networks are called DCNNs but the abbreviations and names between these two are often used exchangeable. Fully connected layers get an input vector containing the image flattened pixels which have been corrected, filtered and reduced using convolution and pooling layers. The softmax function is applied to the outputs of the fully connected layers to give the probability of a class the image belongs to.

However, newly the profit of this has been questioned because many parameters are introduced by this, leading potentially to overfitting. Consequently, many researchers started to construct CNN architecture without fully connected layer using other techniques such as max-over-time pooling to substitute the linear layers role.

П-Nets [14] are a new class of DCNNs, particularly polynomial neural networks with output as high-order polynomial of the input, and can be implemented using special type of skip connections and their parameters can be represented via high-order tensors. The П-Nets have better representation power than typical DCNNs and produce good results without the use of non-linear activation functions in a large tasks number.

The П-Nets make state-of-the-art results in solving such tasks as image generation. In family of П΄Nets the output is a high-order polynomial of the input. For avoiding the combinatorial explosion in the parameters number of polynomial activation functions, П΄Nets use a special skip connections type to implement the polynomial expansion.

Normalisation [15] is the process of subtracting the mean and dividing by the standard deviation. It transforms the range of the data to be between –1 and 1 making the data use the same scale sometimes called Min-Max scaling. It is common to normalize the input features, standardizing the data by removing the mean and scaling to unit variance. It is often important when the input features are centered around zero and have variance of the same order. With some tasks such as image analysis the data are scaled so that its range is between 0 and 1, most simply dividing the pixel values by 255. This also allows the training process to find the optimal parameters quicker.

Batch normalization [15] has benefits of helping to make a network output giving more stable predictions, reduce overfitting through regularization and speed up training by an order of magnitude. Batch normalization is the process of carrying normalization within the scope activation layer of the current batch, subtracting the mean of the batches activations and dividing by the standard deviation of them. This is necessary as even after normalizing the input as some activation can be higher, which can cause the subsequent layers to act abnormally and make the network less stable.

As batch normalization has scaled and shifted the activation outputs, the weights in the next layer will no longer be optimal.

Stochastic gradient descent (SGD) would undo the normalization, as it would minimize the loss function. To prevent this effect, two trainable parameters can be added to each layer to allow SGD to deformalize the output. These parameters are a mean parameter "beta" and a standard deviation parameter "gamma". Batch normalization sets these two weights for each activation output to allow the normalization to be reversed to get the raw input; this avoids affecting the stability of the network by avoiding update the other weights.

Universal approximation theorem [15]. A large enough CNN can solve any solvable problem. The *Universal approximation theorem* essentially states if a problem can be solved it can be solved by a deep neural network, given enough layers of affine functions layered with non-linear functions. Essentially a stack of linear functions followed by non-linear functions could solve any problem that is solvable.

Practically when implementing this, it can be many matrix multiplications with large enough matrices followed by RELU, stacked together these have a mathematical property resulting in being able to solve any arbitrary complex mathematical function to any arbitrary high level of accuracy assuming one have the time and resource to train it.

Whether this would give the neural network understanding is a debated topic, especially by cognitive scientists. The argument is that no matter how well you approximate the syntax and semantics of a problem, you never understand it. This is basically the foundation of Searle's Chinese Room Argument. Some would argue that does it matter if you can approximate the solution to the problem well enough that it is indistinguishable from understanding the problem.

Deep convolutional NNs are mainly focused on applications such as detection of objects, processing of images, processing of audio and natural language processing, image classification, optical character recognition.

It is worth to emfasize that all the deep neural networks mentioned above are those with the given number of layers.

MULTILAYERED ITERATIVE ALGORITHM GMDH AS A SELF-ORGANIZING DEEP NEURAL NETWORK

The worldwide known MIA GMDH [16, 17] is effective data mining tool, one of the most successful inductive modeling methods which extraction knowledge directly from the data based on experimental measurements or statistical observations. It was developed by O. Ivakhnenko in 1965 as multilayer deep feed forward neural network that belongs to class of deep Polynomial Neural Networks with theoretically unlimited number of layers which are set during learning process. It is now considered as the first ever multi-layer self-organized NN and O. Ivakhnenko is often called as the father of deep learning [1, 16]. The GMDH algorithm was first used in modeling complex systems with a number of inputs and one output.

The main GMDH network goal is actually to construct a function in a feed-forward network on the basis of a second-degree transfer function. The method uses information directly from a data sample and minimizes the impact of an author a priori assumptions on the modeling results; finds regularity in data and selects informative input arguments; automatically finds the model structure and its parameters.

Any GMDH algorithm solves a discrete optimization task to construct the optimal complexity model by the given external criterion minimum based on the data sample separation:

$$f^* = \arg\min_{f \in \Phi} CR(f),$$

where CR is a selection criterion as a measure of the model $f \in \Phi$ quality. A model selection criterion is called "external" if it is based on additional information that is not contained in the data used for calculation of model parameters. The set Φ of models being compared can be formed using iterative and sorting-out generators of model structures of diverse complexities which differ by various variants of generation techniques and organization of minimum search of a given external criterion based on additional information that is not contained in the data used for calculation of model parameters.

For realizing the external supplement principle, the parameter estimations by the least squares method LSM (usually) and criteria values are calculated on various parts of a sample W=[X, y], where X, y are a matrix and vector of n measurements of m arguments and one output respectively. The input sample W is splitted into two subsets: training (A) and checking (B). The *regularity criterion* calculated for a model $f \in \Phi$ is most commonly used among all GMDH criteria:

$$AR_{B|A}(f) = \left\| y_B - \hat{y}_{B|A}(f) \right\|^2 = \left\| y_B - X_{Bf} \hat{\theta}_{Af} \right\|^2$$

which means "a model f error on B with parameters obtained on A", and X_{Af}, X_{Bf} are submatrices of the matrix X containing columns that correspond to a partial model $f \in \Phi$ being considered.

One of main elements of an iterative GMDH algorithm such as the polynomial partial description can be considered as an elementary neuron of GMDH neural network. The GMDH NN solves discrete optimization task by successive approaching to the criterion minimum: the models complexity on a layer arises due to the pairwise "crossing" F best models from the previous layer.

The complication process stops after the criterion starts to increase. Originality of the network with such neurons consists in high speed of local adjustment of neuron weights and automatic global optimization of units number. *The depth of* MIA GMDH *network is determined automatically.* The MIA GMDH is used in different problems of knowledge discovery and data analysis, modeling of nonlinear systems, function approximation, forecasting and modeling, classification, pattern recognition and clustering.

Therefore, MIA GMDH is a kind of deep neural networks without given number of layers. Other types of the GMDH-based networks are presented below.

GMDH-BASED DEEP POLYNOMIAL NNs

PNN [18] is a flexible neural architecture whose structure is developed through learning. This network eliminates some GMDH drawbacks such as 1) tending for generation quite complex polynomial for relatively simple systems; 2) tending to produce an overly complex model with highly nonlinear systems; 3) enabling to generate a highly versatile structure with less than three input variables.

The PNN come with a high flexibility level as each node (processing element forming a partial description (PD)) can have a various number of input variables as well as exploit a various order of the polynomial (linear, quadratic, cubic etc.). Architecture is not fixed in advance and becomes fully optimized parametrically and structurally. Particularly, the number of the PNN architecture layers can be modified by adding new layers, if necessary, namely network grows over the training period. In this sense, PNN is a *self-organizing network*. Performance of DPNN depends strongly on the input variables number and the polynomial order which are determined by trial and error approach.

Genetic polynomial neural network (GPNN) [19] is used for improving the PNN performance. GPNN determines the input variables number and the all neurons order with genetic algorithm (GA). Genetic algorithm used for searching between all possible values for the input variables number and the polynomial order. Each of the PD outputs is obtained by using a few kinds of high order polynomials (linear, quadratic, cubic) of the input variables. Although the PNN is structured by a systematic design procedure, it has some drawbacks to be solved. PNN productivity depends on the number of input arguments and the each PD order. These parameters must be determined by trial and error method, which has low efficiency and heavy computational load. The GPNN uses GA to alleviate the above averted drawbacks. The GA is used for determining the input variables number in each PD and to determine the appropriate kind of polynomials in each PD.

There are also many GMDH-type NNs [20] which are characterized by absence of the sample separation into training and checking parts and use various model quality criteria. But they generally retain the MIA GMDH main structure features, particularly selection process.

In general, the hybridization approach has proven to be effective for training neural networks, in particular the GMDH-based NNs.

COMPARATIVE ANALYSIS OF DEEP FEED-FORWARD DETERMINISTIC NEURAL NETWORKS

The main features of complex NNs analysis are following:

1) overfitting,
2) number of layers,
3) ability to dimension reduce,
4) ability to synthesize the models,
5) type of connections between neurons,
6) training method,
7) number of outputs,
8) application areas.

The comparison of these features in MIA GMDH and others deep feed-forward NNs is presented below.

Multilayered perceptron: 1) overfitting: exist; 2) number of layers: fixed; 3) availability of dimension reduce: absent; 4) ability to synthesizes the models: exist; 5) type of connections between neurons: fully connected; 6) learning algorithm: gradient descent; 7) number of outputs: many; 8) main application: classification, modelling, prediction.

Convolutional NNs: 1) overfitting: exist; 2) number of layers: constant; 3) availability of dimension reduce: absent; 4) ability to synthesizes the models: exist; 5) type of connections between neurons: fully connected; 6) learning algorithm: gradient descent; 7) number of outputs: many; 8) application: processing of images, audio, natural language, image classification.

GMDH-based NNs: 1) overfitting: absent; 2) number of layers: dynamic; 3) availability of dimension reduce: exist; 4) ability to synthesizes the models: exist; 5) type of connections between neurons: partly connected; 6) learning algorithm: LMS; 7) number of outputs: one; 8) main application: modeling, forecasting, classification, clasterization, clusterization.

Compared to MLP, CNN-based NNs possesses the following advantages: 1) local connections (each neuron connected only to a little number of neurons for speed up convergence and reducing parameters); 2) sharing of weight (a connections group can share the same weights for reducing parameters); 3) down-sampling reduction of dimensionality.

Neuron networks based on GMDH have the advantages in comparison with convolutional networks by the absence of overfitting, self-tuning of the architecture, the ability to obtain a model in explicit form, but their drawback is the existence of only one possible solution.

Therefore, the main idea of constructing a new class of hybrid deep networks based on MIA GMDH is to create hybrid structures with methods of computational intelligence, in particular genetic algorithms.

Table 1. These comparative features are presented below in the table.

Features	***Multilayered perceptron***	***Convolutional NNs***	***GMDH-based NNs***
Overfitting	present	present	absent
Number of layers	fixed	constant	dynamic
Availability of dimension reduce	absent	exists	exists
Ability to synthesize the models	exists	absent	exists
Type of connections between neurons	fully connected	partly connected	pairwise connected
Learning algorithm	gradient descent	gradient descent	LMS
Number of outputs	multiple	multiple	single
Main application areas	modelling, prediction, classification	processing of images, audio, natural language, images classification	modeling, forecasting, classification, clasterization

CONCLUSION

In this article a comparative analysis of neural networks based on GMDH and the most well-known deep supervised feed-forward neural networks is presented.

Deterministic networks include networks in which the number of layers is constant (extreme learning machines, liquid state machines, convolutional NN and some her modifications), ones with given layers number that can change due to the peculiarity of the problem being solved (multilayered perceptron), and neural networks with a dynamic number of layers that is adjusted by the network itself during the learning process.

The main representative of the latest self-organizing networks, which potentially give the best results, especially in the case of a large number of input data, is MIA GMDH and its modifications such as Deep polynomial neural network, Genetic polynomial neural network and GMDH-type NNs. These NNs do not use gradient-based methods for training and do not suffer from the overfitting.

The main output of this analysis is that hybridization of GMDH-based NN with stochastic methods of computational intelligence is promising to achieve a synergetic effect.

REFERENCES

1. Schmidhuber J. Deep Learning in Neural Networks: An Overview. *Neural Networks*. 2015, Vol. 61, pp. 85–117.
2. Bengio Y. Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning*. 2009, Vol. 2: No. 1, pp. 1–127.
3. Li Deng. A tutorial survey of architectures, algorithms, and applications for deep learning. APSIPA Transactions on Signal and Information Processing, Volume 3, E2, 2014, pp.1-29. doi:10.1017/atsip.2013.9
4. Li Deng, Dong Yu. Deep Learning: Methods and Applications. *Foundations and Trends in Signal Processing*. 2014, Vol. 7: No. 3–4, pp. 197–387.
5. Shahroudnejad A. A Survey on Understanding, Visualizations, and Explanation of Deep Neural Networks. CoRR, 2021, URL: https://arxiv.org/pdf/2102.01792v1.pdf (Last accesed: 10.08.2021)
6. Shaveta D., Munish K., Maruthi Rohit A., Gulshan K. A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning. *Archives of Computational Methods in Engineering*. 2020, 27 (4), pp. 1071–1092.
7. URL: https://www.mdpi.com/2504-3900/47/1/9 (Last accesed: 15.08.2021)
8. Ivakhnenko A.G., Lapa V.G.. Cybernetic Predicting Devices. *CCM Information Corporationm*, 1965, 256 p. (In Russia)
9. Ben-Bright B., Zhan Y., Ghansah B., Amankwah R., Keddy Wornyo D., Ansah E. Taxonomy and a Theoretical Model for Feedforward Neural Networks. *International Journal of Computer Applications*. 2017, 163(4), pp. 39–49.
10. Hopfield J.J. Neural Networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci.* USA, 1982, 79, pp. 2554–2558.
11. Li Z., Yang W., Peng S., Liu F. A Survey of Convolutional Neural Networks. *Analysis, Applications, and Prospects*. 2020, pp.21
12. URL: https://fanchenyou.github.io/homepage/docs/cnn_survey.pdf (Last accessed: 10.07.2021)
13. Srinivas S. et al. A Taxonomy of Deep Convolutional Neural Nets for Computer Vision. *Frontiers in Robotics and AI*. 2016, pp.?
14. Chrysos G., Moschoglou S., Bouritsas G., Deng J., Panagakis Y., Zafeiriou S. Deep Polynomial Neural Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (T-PAMI), 2021.

15. URL: https://towardsdatascience.com/an-introduction-to-convolutional-neural-networks-eb0b60b58fd7 (Last accessed: 17.07.2021)
16. Stepashko V. Developments and Prospects of GMDH-Based Inductive Modeling. *Advances in Intelligent Systems and Computing II: Selected Papers from the International Conference on Computer Science and Information Technologies* CSIT 2017 / N. Shakhovska, V. Stepashko, Editors. AISC book series, Springer, 2018, Vol. 689, pp. 474–491.
17. Stepashko V. On the Self-Organizing Induction-Based Intelligent Modeling. *Advances in Intelligent Systems and Computing III: Selected Papers from the International Conference on Computer Science and Information Technologies* CSIT 2018 / N. Shakhovska, M.O. Medykovskyy, Editors. AISC book series, Springer, 2019, Vol. 871, pp. 433–448.
18. Oh S.-K., Pedrycz W., Park B.-J. Polynomial neural networks architecture: analysis and design. *Computers and Electrical Engineering*. 2003, 23, pp. 703–725.
19. Farzi S. A New Approach to Polynomial Neural Networks based on Genetic Algorithm. *International Scholarly and Scientific Research & Innovation*. 2008, 2(8), pp. 2700–2707.
20. Moroz O.H., Stepashko V.S. An overview of hybrid structures of GMDH-like neural networks and genetic algorithms. *Inductive modeling of complex systems: Coll. sciences works*. 2015,7, K .: IRTC ITS NASU, pp. 173–191.

Received 03.08.2021

Мороз О.Г., канд. техн. наук,
старш. наук. співроб. відд. інформаційних
технологій індуктивного моделювання
ORCID: 0000-0002-0356-8780,
e-mail: olhahryhmoroz@gmail.com
Степашко В.С., д-р. техн. наук, професор
зав. відд. інформаційних технологій
індуктивного моделювання
ORCID: 0000-0001-7882-3208,
e-mail: stepashko@irtc.org.ua
Міжнародний науково-навчальний центр
інформаційних технологій та систем
НАН України та МОН України,
пр. Академіка Глушкова, 40, м. Київ, 03187, Україна

ПОРІВНЯЛЬНІ ОСОБЛИВОСТІ БАГАТОРЯДНОГО ІТЕРАЦІЙНОГО АЛГОРИТМУ МГУА ТА ГЛИБИННИХ НЕЙРОМЕРЕЖ ПРЯМОГО ПОШИРЕННЯ

Вступ. Глибинні нейронні мережі є ефективним інструментом для розв’язання актуальних задач: аналізу даних, моделювання, прогнозування, розпізнавання образів, кластеризації, класифікації тощо. Вони відрізняються між собою зокрема архітектурою, методом навчання. Найпростішими і широко використовуваними є глибинні нейромережі прямого поширення.

Метою статті є коротке порівняння основних особливостей глибинних детермінованих контрольованих нейронних мереж прямого поширення з нейронною мережею багаторядного ітераційного алгоритму (БІА) МГУА та формулювання основних ідей побудови нового класу гібридних глибинних мереж на основі БІА МГУА.

Методи. Було вивчено найбільш використовувані глибинні контрольовані нейронні мережі прямого поширення: багатошаровий персептрон, згорткову нейромережу та деякі її модифікації, поліноміальну нейронну мережу, генетичну поліноміальну нейронну мережу тощо.

Результати. Здійснено порівняльний аналіз основних особливостей нейронної мережі БІА МГУА з характеристиками інших глибинних детермінованих контрольованих нейронних мереж. Визначено найбільш перспективні підходи до вдосконалення продуктивності цієї мережі, зокрема, на основі гібридизації з методами обчислювального інтелекту. Сформульовано основну ідею побудови нового класу гібридних глибинних мереж на основі БІА МГУА.

Висновки. Оригінальним представником самоорганізовних мереж, які потенційно дають кращі результати, особливо в разі великих даних, є БІА МГУА та його модифікації. Запропоновано гібридизацію нейромереж на основі МГУА зі стохастичними методами обчислювального інтелекту для досягнення синергетичного ефекту.

***Ключові слова**: багатошаровий ітераційний алгоритм МГУА (БІА МГУА), самоорганізовна нейронна мережа, архітектура нейронної мережі, глибинні нейронні мережі, нейронні мережі прямого поширення, нейронні мережі з учителем, глибинне навчання.*