Matrix Deep Neural Network and Its Rapid Learning in Data Science Tasks

Iryna Pliss[1], Olena Boiko[2], Valentyna Volkova[3], Yevgeniy Bodyanskiy[4]

1. Control Systems Research Laboratory, Kharkiv National University of Radio Electronics, UKRAINE, Kharkiv, Nauky ave., 14, email: iryna.pliss@nure.ua

2. Control Systems Research Laboratory, Kharkiv National University of Radio Electronics, UKRAINE, Kharkiv, Nauky ave., 14, email: olena.boiko@ukr.net

3. Samsung Electronics Ukraine Company, LLC R&D (SRK), UKRAINE, Kyiv, Lva Tolstogo St., 57, email: v.volkova@samsung.com

4. Control Systems Research Laboratory, Kharkiv National University of Radio Electronics, UKRAINE, Kharkiv, Nauky ave., 14, email: yevgeniy.bodyanskiy@nure.ua

Abstract: **The matrix deep neural network and its learning algorithm are proposed. This system allows reducing the number of tunable weights due to the rejection of the operations of vectorization-devectorization. It also saves the information between rows and columns of 2D inputs.**

Keywords: **deep learning, multilayer network, data mining, 2D network.**

I. Introduction

Nowadays, artificial neural networks (ANNs) are widely used to solve many problems arising in Data Science. Here, multilayer perceptron (MLP) [1,13-18] is the most widely used. On the basis of MLP deep neural networks (DNNs) [2-4,19,21] were developed, that have improved characteristics in comparison with their prototypes, namely traditional shallow neural networks.

In the general case, a multilayer perceptron that contains L information processing layers ($L-1$ hidden layer and one output layer) realizes a nonlinear transformation that can be written in the form

$$\hat{Y}(k)=\Psi(X(k))=\Psi^{[L]}\left(W^{[L]}(k-1)\Psi^{[L-1]}\times\right.$$
$$\left.\times\left(W^{[L-1]}(k-1)\Psi^{[L-2]}\left(\ldots\Psi^{[1]}\left(W^{[1]}(k-1)X(k)\right)\right)\right)\right)$$

where:

- $\hat{Y}(k)$ denotes vector output signal of corresponding dimensions;
- $X(k)$ denotes vector input signal of corresponding dimensions;
- $\Psi^{[l]}$ are diagonal matrices of activation functions on each layer;
- $W^{[l]}(k-1)$ are matrices of synaptic weights that are adjusted during the learning process based on error backpropagation;
- $l=1,2,\ldots,L$;
- $k=1,2,\ldots$ is discrete time index.

In the DNN family, the most popular are the convolutional neural networks (CNNs) [20,22-25] that are mainly designed to process images represented in the form of $(n_1\times n_2)$-matrices $X(k)=\{x_{i_1i_2}(k)\}$ (where $i_1=1,2,\ldots,n_1$ and $i_2=1,2,\ldots,n_2$), which must be vectorized before submission to the network, i. e. they must be presented in the form of vectors [10], the dimension of which can be quite large, that leads to the effect of "curse of dimensionality".

This effect can be avoided by processing the original matrix using convolution, pooling and encoding operations. As a result a vector of dimension smaller than $(n_1n_2\times 1)$ is fed to the perceptron's input.

Although DNNs provide high quality of the information processing, their training time is too long, and the training process itself may require considerable computing resources. However, it is possible to speed up the information processing by bypassing the operations of vectorization-devectorization, i.e. by storing information that will be processed not in the form of a vector, but in the form of a matrix.

The abovementioned problem is solved by the matrix neural networks [5,6,11,12], that are quite complex from the computational point of view.

In this connection, it seems expedient to develop architecture and algorithms for tuning a deep matrix neural network that is characterized by the simplicity of the numerical realization and high speed of its synaptic weights learning.

II. Adaptive Bilinear Model

The proposed matrix DNN is based on the adaptive matrix bilinear model introduced earlier by the authors [7, 8]

$$\hat{Y}(k)=\{\hat{y}_{j_1j_2}\}=A(k-1)X(k)B(k-1),$$
$$j_1=1,2,\ldots,n_1; \qquad (1)$$
$$j_2=1,2,\ldots,n_2$$

where $A(k-1)$, $B(k-1)$ are $(n_1\times n_1)$, $(n_2\times n_2)$-matrices of tunable parameters that are adjusted during online learning-identification process.

For this, either the gradient adaptation procedure

$$\begin{cases} A(k) = A(k-1) + \eta_A(k) \times \\ \qquad \times E(k) B^{\mathrm{T}}(k-1) X^{\mathrm{T}}(k), \\ B(k) = B(k-1) + \eta_B(k) \times \\ \qquad \times X^T(k-1) A^T(k) E_A(k) \end{cases} \tag{2}$$

is used or its version optimized by speed [7] that can be written as

$$\begin{cases} A(k) = A(k-1) + \big(Tr\ E(k) B^{\mathrm{T}}(k-1) \times \\ \quad \times X^{\mathrm{T}}(k) X(k) B(k-1) E^{\mathrm{T}}(k)\big) \times \\ \quad \times \big(Tr\ E(k) B^{\mathrm{T}}(k-1) X^{\mathrm{T}}(k) X(k) \times \\ \quad \times B(k-1) B^{\mathrm{T}}(k-1) X^{\mathrm{T}}(k) X(k) \times \\ \quad \times B(k-1) E^{\mathrm{T}}(k)\big)^{-1} E(k) \times \\ \quad \times B^{\mathrm{T}}(k-1) X^{\mathrm{T}}(k), \\ B(k) = B(k-1) + \big(Tr\ E_A^{\mathrm{T}}(k) A(k) X(k) \times \\ \quad \times X^{\mathrm{T}}(k) A^{\mathrm{T}}(k) E_A(k)\big)\big(Tr\ A(k) \times \\ \quad \times X(k) X^{\mathrm{T}}(k) A^{\mathrm{T}}(k) E_A(k) E_A^{\mathrm{T}}(k) \times \\ \quad \times A(k) X(k) X^{\mathrm{T}}(k) A^{\mathrm{T}}(k)\big)^{-1} \times \\ \quad \times X^{\mathrm{T}}(k-1) A^{\mathrm{T}}(k) E_A(k), \end{cases} \tag{3}$$

that is the matrix generalization of the Kaczmarz–Widrow–Hoff learning algorithm (here $\eta_A(k)$, $\eta_B(k)$ are learning rate parameters,

$$\begin{cases} E(k) = Y(k) - A(k-1) X(k) B(k-1), \\ E_A(k) = Y(k) - A(k) X(k) B(k-1), \end{cases}$$

$Y(k)$ is reference matrix signal).

The learning algorithm in Eq. (3) can be given additional filtering properties if the learning rate parameters in Eq. (2) are calculated using the recurrence relations that can be written in the form

$$\begin{aligned} \eta_A^{-1}(k) = r_A(k) = \beta r_A(k-1) + \\ + Tr\big(E(k) B^{\mathrm{T}}(k-1) \times \\ \times X^{\mathrm{T}}(k) X(k) B(k-1) \times \\ \times B^{\mathrm{T}}(k-1) X^{\mathrm{T}}(k) X(k) \times \\ \times B(k-1) E^{\mathrm{T}}(k)\big) \end{aligned}$$

and

$$\begin{aligned} \eta_B^{-1}(k) = r_B(k) = \beta r_B(k-1) + \\ \times Tr\big(A(k) X(k) X^{\mathrm{T}}(k) \times \\ \times A^{\mathrm{T}}(k) E_A(k) E_A^{\mathrm{T}}(k) A(k) \times \\ \times X(k) X^{\mathrm{T}}(k) A^{\mathrm{T}}(k)\big) \end{aligned}$$

where $0 \le \beta \le 1$ is smoothing parameter [9].

On the basis of the model from Eq. (1), it is easy to introduce its nonlinear modification that can be written in the following form:

$$\begin{aligned} \hat{Y}(k) = \{\hat{y}_{j_1 j_2}\} = \Psi \odot \big(A(k-1) X(k) B(k-1)\big) = \\ = \Psi \odot U(k), \end{aligned} \tag{4}$$

which is in fact the matrix generalization of the transformation that is realized by any of the layers of a multilayer perceptron.

In Eq. (4) Ψ denotes a $(n_1 \times n_2)$-matrix of activation functions, that acts elementwise on the matrix of internal activation signals of the system that are denoted by $U(k) = \{u_{j_1 j_2}(k)\}$.

In this case, the adjustment of the parameters of the nonlinear matrix model in Eq. (4) can be realized on the basis of the modified δ-rule

$$\begin{cases} a_{j_1 j_2}(k) = a_{j_1 j_2}(k-1) + \eta_A(k) e_{j_1 j_2}(k) \times \\ \quad \times \psi'\big(u_{j_1 j_2}(k)\big) \sum_{i_2=1}^{n_2} b_{j_1 j_2}(k-1) x_{i_1 i_2}(k) = \\ \quad = a_{j_1 j_2}(k-1) + \eta_A(k) e_{j_1 j_2}(k) \times \\ \quad \times \psi'\big(u_{j_1 j_2}(k)\big) \hat{x}_{i_1}(k) = a_{j_1 j_2}(k-1) + \\ \quad + \eta_A(k) \delta_{j_1 j_2}(k) \hat{x}_{i_1}(k), \\ b_{j_1 j_2}(k) = b_{j_1 j_2}(k-1) + \eta_B(k) e_{A\, j_1 j_2}(k) \times \\ \quad \times \psi'\big(u_{A\, j_1 j_2}(k)\big) \sum_{i_1=1}^{n_1} a_{j_1 j_2}(k-1) x_{i_1 i_2}(k) = \\ \quad = b_{j_1 j_2}(k-1) + \eta_B(k) e_{A\, j_1 j_2}(k) \times \\ \quad \times \psi'\big(u_{A\, j_1 j_2}(k)\big) \hat{x}_{i_2}(k) = b_{j_1 j_2}(k-1) + \\ \quad + \eta_B(k) \delta_{A\, j_1 j_2}(k) \hat{x}_{i_2}(k). \end{cases} \tag{5}$$

On the basis of Eq. (4) it is easy to introduce into consideration a multilayer matrix neural network that realizes the transformation

$$\begin{aligned} \hat{Y}(k) = \Psi \odot \Big(A^{[L]}(k-1)\Big(\Psi \odot \Big(A^{[L-1]}(k-1) \times \\ \times \Big(\ldots \Psi \odot \Big(A^{[1]}(k-1) X(k) B^{[1]}(k-1)\Big) \ldots\Big) \times \\ \times B^{[L-1]}(k-1)\Big)\Big) B^{[L]}(k-1)\Big) \end{aligned} \tag{6}$$

Using the learning algorithm from Eq. (5) and error backpropagation, it is possible to obtain the adaptive procedure for tuning all parameters of the matrix DNN in Eq. (6):

- for the output layer:

$$\begin{cases} a_{j_1 j_2}^{[L]}(k) = a_{j_1 j_2}^{[L]}(k-1) + \eta_A(k) \delta_{j_1 j_2}^{[L]}(k) \hat{o}_{i_1}^{[L-1]}(k), \\ b_{j_1 j_2}^{[L]}(k) = b_{j_1 j_2}^{[L]}(k-1) + \eta_B(k) \delta_{A\, j_1 j_2}^{[L]}(k) \hat{o}_{A\, i_2}^{[L-1]}(k) \end{cases}$$

where

$$\delta_{j_1 j_2}^{[L]}(k) = \psi'\Big(u_{j_1 j_2}^{[L]}(k)\Big) e_{j_1 j_2}(k),$$

$$\hat{o}_{i_1}^{[L-1]}(k) = \sum_{i_2=1}^{n_2} b_{j_1 j_2}^{[L]}(k-1) o_{i_1 i_2}^{[L-1]}(k),$$

$$\delta_{A\, j_1 j_2}^{[L]}(k) = \psi'\Big(u_{A\, j_1 j_2}^{[L]}(k)\Big) e_{A\, j_1 j_2}(k),$$

$$\hat{o}^{[L-1]}_{A\,i_2}(k)=\sum_{i_1=1}^{n_1} a^{[L]}_{j_1 j_2}(k)\, o^{[L-1]}_{i_1 i_2}(k);$$

- for the l th hidden layer, $1<l<L$:

$$\begin{cases} a^{[l]}_{j_1 j_2}(k)=a^{[l]}_{j_1 j_2}(k-1)+\eta_A(k)\,\delta^{[l]}_{j_1 j_2}(k)\,\hat{o}^{[l-1]}_{i_1}(k), \\ b^{[l]}_{j_1 j_2}(k)=b^{[l]}_{j_1 j_2}(k-1)+\eta_B(k)\,\delta^{[l]}_{A\,j_1 j_2}(k)\,\hat{o}^{[l-1]}_{A\,i_2}(k) \end{cases}$$

where

$$\delta^{[l]}_{j_1 j_2}(k)=\psi'\left(u^{[l]}_{j_1 j_2}(k)\right)\sum_{i_1=1}^{n_1}\delta^{[l+1]}_{j_1 j_2}a^{[l+1]}_{j_1 j_2}(k),$$

$$\hat{o}^{[l-1]}_{i_1}(k)=\sum_{i_2=1}^{n_2} b^{[l]}_{j_1 j_2}(k-1)\, o^{[l-1]}_{i_1 i_2}(k),$$

$$\delta^{[l]}_{A\,j_1 j_2}(k)=\psi'\left(u^{[l]}_{A\,j_1 j_2}(k)\right)\sum_{i_2=1}^{n_2}\delta^{[l+1]}_{A\,j_1 j_2}(k)\, b^{[l+1]}_{j_1 j_2}(k),$$

$$\hat{o}^{[l-1]}_{A\,i_2}(k)=\sum_{i_1=1}^{n_1} a^{[l]}_{j_1 j_2 i_1}(k)\, o^{[l-1]}_{i_1 i_2}(k);$$

- for the first hidden layer:

$$\begin{cases} a^{[1]}_{j_1 j_2}(k)=a^{[1]}_{j_1 j_2}(k-1)+\eta_A(k)\,\delta^{[1]}_{j_1 j_2}(k)\,\hat{o}^{[0]}_{i_1}(k), \\ b^{[1]}_{j_1 j_2}(k)=b^{[1]}_{j_1 j_2}(k-1)+\eta_B(k)\,\delta^{[1]}_{A\,j_1 j_2}(k)\,\hat{o}^{[0]}_{A\,i_2}(k) \end{cases}$$

where

$$\delta^{[1]}_{j_1 j_2}(k)=\psi'\left(u^{[1]}_{j_1 j_2}(k)\right)\sum_{i_1=1}^{n_1}\delta^{[2]}_{j_1 j_2}a^{[2]}_{j_1 j_2}(k),$$

$$\hat{o}^{[0]}_{i_1}(k)=\sum_{i_2=1}^{n_2} b^{[1]}_{j_1 j_2}(k-1)\, x_{i_1 i_2}(k),$$

$$\delta^{[1]}_{A\,j_1 j_2}(k)=\psi'\left(u^{[1]}_{j_1 j_2}(k)\right)\sum_{i_2=1}^{n_2}\delta^{[2]}_{A\,j_1 j_2}(k)\, b^{[2]}_{j_1 j_2}(k),$$

$$\hat{o}^{[0]}_{A\,i_2}(k)=\sum_{i_1=1}^{n_1} a^{[1]}_{j_1 j_2}(k)\, x_{i_1 i_2}(k).$$

III. COMPUTATIONAL EXPERIMENTS

The efficiency of the proposed system and learning methods was demonstrated on the classification task. A number of experiments was carried out on the MNIST dataset that was introduced by Yann LeCun and Corinna Cortes [26].

This dataset is widely used for training and testing in machine learning, namely in the classification task. This dataset contains 60000 training observations and 10000 test observations.

Each observation is an image of size 28x28 pixels that represents a handwritten digit. In general the dataset has 10 classes (digits from 0 to 9).

Some examples of the images from this dataset are presented in Fig. 1.

The elements of an image are represented by pixel values from 0 to 255, where 0 means white pixel (background) and 255 means black pixel (foreground). These values were preprocessed before training using normalization. The inputs for the network were $(n_1\times n_2)$-matrices, where $n_1=n_2=28$. Every hidden layer also had size of $n_1\times n_2=28\times 28$.

The results of the computational experiments are presented in Table 1.

TABLE 1. EXPERIMENTAL RESULTS

Number of layers in the network	Error on test set, %
3	25
5	20
10	18

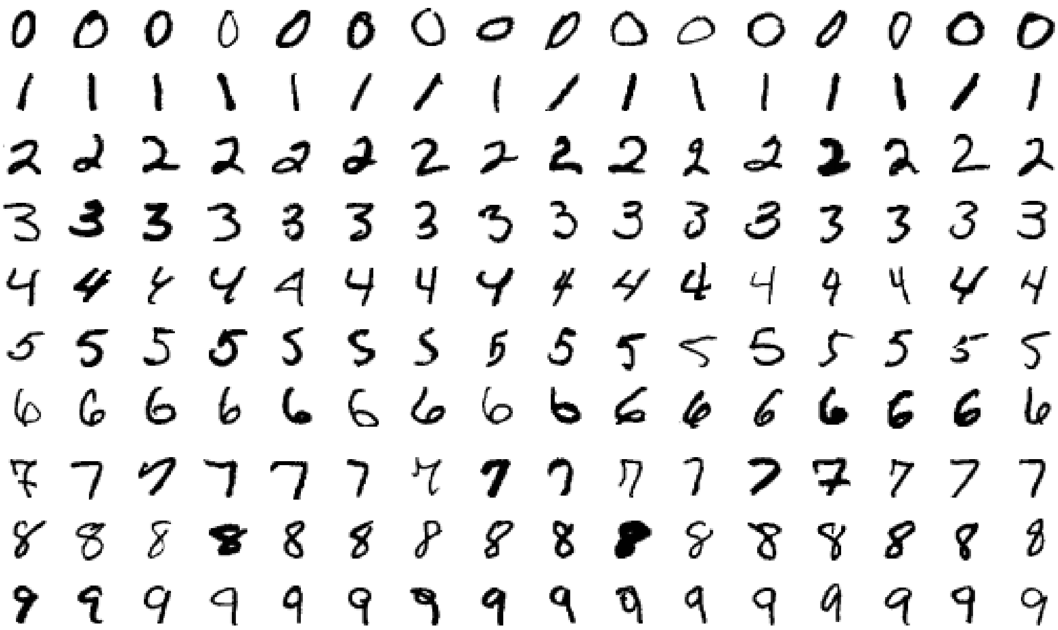

Fig.1. Examples of the images from the MNIST dataset.

IV. Conclusion

In this paper the matrix deep neural network and its learning algorithm are proposed. They allow significantly to reduce the number of adjustable weights due to the rejection of the vectorization-devectorization operations of 2D input signals.

One of the main advantages of the proposed system is that it also preserves the information between rows and columns of 2D inputs of the system.

The considered DNN in comparison with traditional multilayer perceptrons has increased speed, determined by reduced number of adjustable parameters and optimization of the learning algorithm, and the simplicity of numerical implementation.

The proposed system can be used to solve a wide range of machine learning tasks, particularly connected with the problems of image processing, where input signals are presented to the system for data processing in the form of a matrix.

References

[1] C. M. Bishop, Neural Networks for Pattern Recognition. Oxford : *Clarendon Press*, 1995.

[2] Y. LeCun, Y. Bengio, G. Hinton, "Deep Learning," *Nature*, vol. 521, pp. 436-444, 2015.

[3] J. Schmidhuber, "Deep Learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85-117, 2015.

[4] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. *MIT Press*, 2016.

[5] P. Daniušis and P. Vaitkus, "Neural networks with matrix inputs," *Informatica*, 19, №4, pp. 477-486, 2008.

[6] J. Gao, Y. Guo, and Z. Wang, "Matrix neural networks," in *Proceedings of the14th International Symposium on Neural Networks (ISNN), Part II*, Sapporo, Japan, 2017, pp. 1–10.

[7] Ye. V. Bodyanskiy, I. P. Pliss, and V. A. Timofeev, "Discrete adaptive identification and extrapolation of two-dimensional fields," *Pattern Recognition and Image Analysis, 5, №3*, pp. 410-416, 1995.

[8] S. Haykin, Neural Networks: A Comprehensive Foundation. Upper Saddle River, *N. J. : Prentice Hall*, Inc., 1999.

[9] S. Vorobyov, Ye. Bodyanskiy, "On a non-parametric algorithm for smoothing parameter control in adaptive filtering," *Engineering Simulation*, vol. 16, p. 314-320, 1999.

[10] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, "Deep learning for visual understanding: A review," *Neurocomputing, vol. 187*, pp. 27-48, 2016.

[11] P. Stubberud, "A vector matrix real time backpropagation algorithm for recurrent neural networks that approximate multi-valued periodic functions," *International Journal on Computational Intelligence and Application, 8(4)*, pp. 395-411, 2009.

[12] M. Mohamadian, H. Afarideh, and F. Babapour, "New 2D Matrix-Based Neural Network for Image Processing Applications," *IAENG (International Association of Engineers) International Journal of Computer Science, 42(3)*, pp. 265-274, 2015.

[13] K. Suzuki, Artificial Neural Networks: Architectures and Applications. NY: *InTech*, 2013.

[14] K. L. Du and M. Swamy, Neural Networks and Statistical Learning. *Springer-Verlag* London, 2014.

[15] D. Graupe, Principles of Artificial Neural Networks (Advanced Series in Circuits and Systems). Singapore: *World Scientific Publishing Co. Pte. Ltd*., 2007.

[16] L. Rutkowski, Computational intelligence. Methods and techniques, Berlin-Heidelberg: *Springer-Verlag*, 2008.

[17] R. Kruse, C. Borgelt, F. Klawonn, C. Moewes, M. Steinbrecher, and P. Held, Computational intelligence, Berlin: *Springer*, 2013.

[18] D. T. Pham and X. Liu, Neural Networks for Identification, Prediction and Control, London: *Springer-Verlag*, 1995.

[19] I. Arel, D. Rose, and T. Karnowski, "Deep machine learning – a new frontier in artificial intelligence research," *IEEE Computational Intelligence Magazine, vol. 5, no. 4*, pp. 13-18, 2010.

[20] K. Kavukcuoglu, P. Sermanet, Y-L. Boureau, K. Gregor, M. Mathieu, Y.. LeCun, "Learning Convolutional Feature Hierachies for Visual Recognition," in *Proceedings of the 23rd International Conference on Neural Information Processing Systems, vol. 1*, pp. 1090-1098, 2010.

[21] C. Dan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3642-3649, 2012.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS'12), vol. 1*, pp. 1097-1105, 2012.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770-778 2016.

[24] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 253-256, 2010.

[25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-9, 2015.

[26] http://yann.lecun.com/exdb/mnist/