UDC 519.7
DOI https://doi.org/10.24144/2616-7700.2025.46(1).166-177

A. Y. Vasko[1], A. Y. Bryla[2]

[1] Uzhhorod National University,
PhD student of Systems Analysis and Optimization Theory
oleksandr.vasko@uzhnu.edu.ua
ORCID: https://orcid.org/0009-0006-1527-505X

[2] Uzhhorod National University,
Associate Professor of Systems Analysis and Optimization Theory,
Candidate of Physical and Mathematical Sciences, Associate Professor
andrii.bryla@uzhnu.edu.ua
ORCID: https://orcid.org/0000-0003-2518-9877

ADAPTIVE LEARNING RATE FOR CNNMVN

Adaptive learning rate techniques are widely used to enhance the training efficiency of neural networks by improving convergence speed and accuracy while decreasing the risk of getting stuck in local minima or saddle points. In this paper, we introduce adaptive learning rate approaches for the Complex-Valued Convolutional Neural Network with Multi-Valued Neurons (CNNMVN), a fully complex-valued neural network that procesoses complex inputs using complex weights and complex-valued activation functions. Unlike traditional real-valued neural networks, CNNMVN employs an error-sharing principle rather than gradient-based optimization, eliminating the local minima problem and allowing for more flexible adjustments in the learning rate.

We propose two adaptive learning rate (ALR) strategies tailored for CNNMVN. The first approach modifies the learning rate coefficients in the error-correction formulas, while the second one adjusts the normalization parameters in the error-backpropagation and error-correction processes. Both methods dynamically adapt the learning rate based on validation accuracy.

Results indicate that adaptive learning rate significantly improves convergence speed and accuracy, particularly when combined with a self-adaptive learning rate. Furthermore, our study highlights the impact of normalization factors on learning dynamics and explores training scenarios where normalization is minimized or removed entirely.

Our findings demonstrate that ALR methods enhance CNNMVN training performance, providing a robust framework for optimizing learning rates in complex-valued neural networks.

Keywords: convolutional neural networks, complex-valued networks, multivalued neurons, CNNMVN, MLMVN, image recognition, frequency domain.

1. Introduction. Deep learning has experienced significant progress with the development of advanced neural network architectures and training methodologies. One of such architectures is the Convolutional Neural Network with Multi-Valued Neuron (CNNMVN). It was first introduced in [1], its modified learning algorithm was presented in [2] and its further deep analysis was presented in [3]. CNNMVN retains the traditional CNN topology while being capable of directly processing complex-valued data, making it particularly useful for applications requiring the handling of magnitude and phase relationships, such as signal processing, radar imaging, and medical diagnostics.

An essential factor influencing the performance of CNNMVN is the choice of the learning rate, which directly impacts convergence speed and generalization capability. In real-valued neural networks various adaptive learning rate techniques

have been proposed to improve training efficiency, such as AdaGrad [4], RMSprop [5], Adam [6], and AdaBelief [7]. These algorithms dynamically adjust the learning rate based on past gradient information. There also is adaptive learning rate techniques based on model complexity [8] and training loss [9]. Similarly, in the domain of complex-valued neural networks, adaptive learning rate techniques have been explored, including complex-valued extensions of Adam

Recent advancements in adaptive learning rate techniques for complex-valued neural networks have introduced several novel approaches aimed at improving convergence efficiency and stability. One such approach is the complex-valued extension of the Adam optimization algorithm [10], which adapts the well-established Adam method to the complex domain, allowing for more effective gradient-based learning in complex-valued neural networks. Another significant contribution is the Adaptive Orthogonal Gradient Descent (AOGD) algorithm [11], specifically designed for fully complex-valued neural networks. The Complex-Valued Adaptive Learning Rate Tuning (CALRT) algorithm [12] has also been proposed to dynamically determine the optimal complex-valued step size (or learning rate) at each iteration of the training process. Additionally, a training algorithm with a selectable search direction for complex-valued feedforward neural networks has been introduced [13]. This method allows for adaptive modifications in the search direction, improving learning efficiency and reducing the risk of stagnation during training.

CNNMVN is a fully complex-valued neural network: its inputs, weights, and activation functions are all complex-valued. Unlike traditional gradient-based methods, CNNMVN learning algorithm is based on the error-sharing principle. This fundamental distinction allows CNNMVN to avoid issues related to local minima, which are commonly encountered in traditional complex-valued gradient descent approaches. As a result, adaptive learning rate algorithms designed for real-valued neural networks (RVNNs) are not suitable for CNNMVN. In this paper, we present several approaches for adaptive learning rates that enhance accuracy and improve convergence speed.

This paper is structured as follows. In Section 2 and 3, we recall some fundamentals related to the error-backpropagation and error-correction algorithms adapted for CNNMVN. Section 4 explores various adaptive learning rate approaches for CNNMVN. In Section 5, we evaluate the effectiveness of these approaches through experimental results on benchmark datasets, demonstrating improvements in convergence speed and model performance. Finally, Section 6 provides conclusions and outlines potential directions for future research in adaptive learning strategies for CNNMVN.

2. CNNMVN error backpropagation process. As it was said before, the learning algorithm in CNNMVN is based on the error-sharing principle [14]. It means that the error produced by each neuron should be proportionally distributed among the neurons connected to it.

Let us briefly recall some fundamentals about CNNMVN error backpropagation [2].

Let us have CNNMVN with G convolutional layers, $M-1$ hidden layers and one output layer. Let each convolutional layer consists of H_g kernels $g = \overline{1,G}$. Let the input image x (or a feature map in the case of convolutional layers from the 2^{nd} one to the G^{th} one) be of size $a_g \times b_g \times d_g$. Then the kernel size in g^{th} convolutional

layer is of size $r_g \times r_g \times d_g$ ($r_g < \min(a_g, b_g)$). The m^{th} hidden layer in the fully connected part contains N_m neurons ($m = \overline{1, M-1}$), and the output layer contains N_M neurons.

Now the global errors of the network δ^*_{nM} should be calculated as

$$\delta^*_{nM} = D_{nM} - Y_{nM}, \quad n = \overline{1, N_M}. \tag{1}$$

The global error then should be shared among all neurons that participate in its formation. Thus, the errors of the output layer neurons are

$$\delta_{nM} = \frac{1}{N_{M-1}+1}\delta^*_{nM}, \quad n = \overline{1, N_M}. \tag{2}$$

To backpropagate the output neurons errors for the neurons in hidden layers we should use the following rule:

$$\delta_{im} = \frac{1}{q_{m-1}} \sum_{n=1}^{N_{m+1}} \delta_{n,m+1}\left(w_i^{n,m+1}\right)^{-1}, \quad i = \overline{1, N_m}, \quad m = \overline{1, M-1}, \tag{3}$$

where δ_{im} is the error of the i^{th} neuron in the m^{th} layer and the normalization factor $q_{m-1} = N_{m-1} + 1$ ($m = \overline{2, M-1}$) is the number of all neurons in the $m-1^{st}$ layer plus one (note that for the first layer ($m = 1$) we use $q_1 = 1$ because there are no preceding layers and no neurons to share the error with). The errors of the first hidden layer neurons are

$$\delta_{i1} = \frac{1}{F} \sum_{n=1}^{N_2} \delta_{n,2}\left(w_i^{n,2}\right)^{-1}, \quad i = \overline{1, N_1}, \tag{4}$$

where F stands for the number of inputs which is equal to the length of flattened feature maps.

Then the errors of the neurons in the first hidden layer should be backpropagated to the last convolutional layer kernels as

$$\delta_{ij}^{hG} = \frac{1}{Q_G} \sum_{n=1}^{N_1} \delta_{n,1}\left(w_{ijh}^{n,1}\right)^{-1}, \quad \begin{array}{c} i = 1, \ldots, a_G - r_G + 1, \\ j = 1, \ldots, b_G - r_G + 1, \\ Q_G = r_G \cdot r_G \cdot c_G, \end{array} \tag{5}$$

where δ_{ij}^{hG} is the error of the feature map pixel with coordinates i, j, produced by the h^{th} kernel of the last (G^{th}) convolutional layer; $\delta_{n,1}$ are the errors of the first hidden layer neurons and $w_{ijh}^{n,1}$ are these neurons' weights that are responsible for processing the ijh^{th} input; Q_G is the normalization factor which should be equal to the size of the kernel in the last convolutional layer.

Now the errors of the kernels in the $g-1^{st}$ convolutional layer are:

$$\delta_{ij}^{l,g-1} = \frac{1}{Q_{g-1}} \sum_{h=1}^{H_g} \sum_{t=1}^{T_g} \delta^{h,g,t}\left(w_t^{h,g}\right)^{-1}, \quad \begin{array}{c} i = 1, \ldots, a_g - r_g + 1, \\ j = 1, \ldots, b_g - r_g + 1, \\ Q_g = r_g \cdot r_g \cdot c_g, \end{array} \tag{6}$$

where $\delta^{l,g-1}$ are the errors of the feature map obtained by the l^{th} kernel in the $g-1^{st}$ convolutional layer ($l = \overline{1, H_{g-1}}$); $\delta^{h,g,t}$ and $w_t^{h,g}$ are the errors of the output and the

weights of the h^{th} kernel in the g^{th} convolutional layer that process the ij^{th} pixel in the t^{th} convolution respectively; Q_{g-1} is a normalization factor which is equal to the size of the kernel in the $g-1^{th}$ layer (note that $Q_2 = 1$ as there is no further error backpropagation from the first convolutional layer).

In [3] were also introduced the error backpropagation rules for convolutional layers with self-adaptive learning rate. Hence, the feature map errors of the last convolutional layer (5) with additional normalization can be found as follows:

$$\delta_{ij}^{hG} = \frac{1}{Q_G \left| c_{ij}^{h,G} \right|} \sum_{n=1}^{N_1} \delta_{n,1} \left(w_{ijh}^{n,1} \right)^{-1}, \quad \begin{array}{c} i = 1, \ldots, a_g - r_g + 1, \\ j = 1, \ldots, b_g - r_g + 1, \\ Q_G = r_G \cdot r_G \cdot c_G, \end{array} \tag{7}$$

and the errors of the feature maps in preceding convolutional layers (6) with additional normalization factor are

$$\delta_{ij}^{l,g-1} = \frac{1}{Q_{g-1} \left| c_{ij}^{h,g-1} \right|} \sum_{h=1}^{H_g} \sum_{t=1}^{T_g} \delta^{h,g,t} \left(w_t^{h,g} \right)^{-1}, \quad \begin{array}{c} i = 1, \ldots, a_g - r_g + 1, \\ j = 1, \ldots, b_g - r_g + 1, \\ Q_g = r_g \cdot r_g \cdot c_g, \end{array} \tag{8}$$

where $\left| c_{ij}^{h,g} \right|$ is the absolute value of the convolved pixel with coordinates i, j in the h^{th} kernel in the g^{th} convolutional layer ($h = \overline{1, H_g}, g = \overline{2, G}$).

3. CNNMVN Error correction. The error correction in CNNMVN should be done in the same manner as suggested in [1] and [3]. The error correction for convolutional layer kernels implies the same idea of the batch LLS-based learning algorithm [15] for MLMVN and the error correction for fully-connected layers is identical to the error correction rule in MLMVN.

Let us have an $a \times b \times d$ input image and a kernel whose size is $r \times r \times d$ ($r < \min(a, b)$). Let K be the number of the convolutional windows in an image through which a convolutional kernel is sliding. Thus $K = (a - r + 1) \times (b - r + 1)$. Each convolutional window can be flattened and represented as an input vector to a kernel. Then, using a matrix-vector notation the convolutional operation can be represented as

$$\begin{bmatrix} x_1 \\ \vdots \\ x_K \end{bmatrix} \begin{bmatrix} w_{1,1,1} \\ \vdots \\ w_{r,r,d} \end{bmatrix} = \begin{bmatrix} z_1 \\ \vdots \\ z_K \end{bmatrix} \tag{9}$$

where x_i is the flattened input vector ($i = \overline{1, K}$); w are the kernel weights and z_i are the convolved pixels. According to a batch algorithm and as it was shown in [1], after calculating the errors for all z_i we can represent adjustments, which should be added to the weights to correct them and obtain the following

$$\begin{bmatrix} x_1 \\ \vdots \\ x_K \end{bmatrix} \begin{bmatrix} \Delta w_{1,1,1} \\ \vdots \\ \Delta w_{r,r,d} \end{bmatrix} = \begin{bmatrix} \delta_1 \\ \vdots \\ \delta_K \end{bmatrix} \tag{10}$$

where $\Delta w_{u,v,j}$ are the adjustments terms for kernel weights ($u, v = \overline{1, r}$ and $j = \overline{1, d}$) and δ_i are the respective errors to be corrected ($i = \overline{1, K}$). We can rewrite (8) as

$$X \cdot \Delta w = \delta, \tag{11}$$

where X consists of K flattened inputs. Thus, system of equations (8) (or (9), which is the same) for unknowns $(\Delta w_{1,1,1}, \ldots, \Delta w_{r,r,d})$ which is typically overdetermined $(K \gg r \cdot r \cdot d)$ can be solved in a similar way

$$\Delta w = X^{\dagger} \cdot \delta, \tag{12}$$

where $X^{\dagger}$ is the Moore-Penrose inversion of X, that is a pseudoinverse matrix of X. Finally, the adjusted weights of the kernel are

$$\tilde{W} = W + C \cdot \Delta w, \tag{13}$$

where W and $\tilde{W}$ are the weighting vectors before and after correction, «+» is a component-wise addition, and C stands for learning rate.

When the wights of the kernels in convolutional layers are adjusted, we should to adjust the errors in the fully-connected layers. It should be done as follows:

$$\tilde{W} = W + \frac{C}{p+1}\delta X^{*}. \tag{14}$$

And the rule with a self-adaptive learning rate is

$$\tilde{W} = W + \frac{C}{(p+1)\,|z|}\delta X^{*}, \tag{15}$$

where W and $\tilde{W}$ are the weighting vectors before and after correction, «+» denotes component-wise addition, C is the learning rate (in general it should be a complex number, but in all practical applications it is usually equal to 1), X^{*} is the vector of reciprocal inputs, p is the number of inputs, and $|z|$ stands for the absolute value of the weighted sum before adjustment (this is a self-adaptive part of the learning rate, and it is significant in MLMVN learning [14, 16]).

4. Adaptive learning rate. Adaptive learning rate (ALR) techniques play a crucial role in real-valued neural networks (RVNNs) by dynamically adjusting the learning rate during training. The primary objectives of ALR in RVNNs include increasing the learning rate to accelerate convergence, reducing training time, and mitigating issues related to local minima and saddle points. By optimizing the learning rate, these networks enhance their training efficiency and overall performance.

However, unlike RVNNs, the learning rule in CNNMVN is not formulated as an optimization problem. Instead, it follows a unique "error-sharing principle," which distributes the error among connected neurons rather than relying on gradient-based updates. As a result, CNNMVN does not depend on traditional gradient descent methods for weight adjustments, making standard ALR techniques designed for RVNNs unsuitable.

A key advantage of CNNMVN's learning rule is that it inherently avoids the problems of local minima and saddle points. Since the learning process does not involve an optimization-based approach, there are no constraints on increasing the learning rate because there is no risk of overshooting or jumping over the minima of an optimization function. This flexibility allows CNNMVN to achieve faster convergence without the risk of getting trapped in undesirable minima, a common issue in gradient-based learning frameworks.

Despite this advantage, the error-sharing principle introduces a challenge when backpropagating through layers with a large number of neurons or inputs. In such cases, the distributed errors become significantly smaller as they propagate through the network. This reduction in error magnitude leads to very small weight adjustment terms, ultimately slowing down the learning process in deeper or more complex network structures.

To address this issue, an adaptive learning rate mechanism can be incorporated into CNNMVN to accelerate training in layers where error values are particularly small. By dynamically increasing the learning rate in these layers, the network can counteract the decreasing weight updates and maintain a more efficient training process. This approach ensures that CNNMVN benefits from faster convergence while preserving the advantages of its non-gradient-based learning rule.

We have designed two adaptive learning rate rules for CNNMVN to enhance training efficiency. The algorithm for both ALR approaches follows these steps. Initially, the learning rate is set to 1. After every K learning samples, we perform a model validation step. Based on the validation accuracy, the learning rate is adjusted accordingly.

The first ALR rule focuses on reducing the normalization factors in the error-backpropagation (2)–(6) and the error-correction formulas (11)–(13). Similar to the first approach, this method maintains a uniform learning rate across all layers but specifically targets the normalization factors as an additional multiplier C in the denominator. In such a case decreasing the learning rate C leads to decrease of the normalization factors. Hence, the network effectively amplifies weight adjustments in layers where the error values are significantly small, thereby accelerating the learning process. The adaptive learning rate coefficients for this approach are determined as

$$C = 4^{4\cdot\left(\frac{\#\ of\ Mismatched\ samples}{\#\ of\ All\ samples}\ -\ 1\right)}. \tag{16}$$

The second approach involves adjusting the learning rate coefficient within the error-correction formulas (12), (13), and (14). In this method, the learning rate remains consistent across all layers, ensuring a uniform update mechanism throughout the network. The learning rate coefficient can vary within a pre-determined range $[1, A]$ and can be adjusted as follows

$$C = A{+}1{-}A \cdot 4^{4\cdot\left(\frac{\#\ of\ Mismatched\ samples}{\#\ of\ All\ samples}\ -\ 1\right)}. \tag{17}$$

Both approaches aim to reduce the slowdown caused by the error-sharing principle, ensuring that CNNMVN achieves faster convergence while maintaining its unique learning framework.

5. Simulation results. To evaluate the effectiveness of the proposed adaptive learning rate methods, we performed experiments using the MNIST dataset of handwritten digits [17]. The dataset consists of 60 000 training and 10 000 testing samples 28×28 pixels each. The network architecture consisted of a simple topology with one convolutional layer containing 16 kernels of size 5×5, followed by a hidden layer with 256 neurons and an output layer with 10 neurons. The soft margins border was equal to $\pi/8$. The training process lasted for 10 epochs. All experiments were performed with the same random seed to compare the dynamic of the learning process.

During training, validation was performed every 1 000 training samples, and the learning rate was adjusted accordingly based on validation accuracy. This ensured that the learning rate dynamically adapted to improve training efficiency.

We provide graphical results demonstrating the testing accuracy for different configurations. Figure 1 presents the learning process with default learning rates and normalization factors as defined by (1)–(6), alongside the same learning process with the self-adaptive (ABS) normalization in the convolutional layers determined by (7) and (8). The results indicate that ABS normalization for the convolutional layer does not significantly impact learning accuracy or convergence time.

To evaluate the first type of adaptive learning rate, we analyzed the default normalization factors of the neural network and identified two key normalization factors that influence the learning process. Based on the model topology and input image size, the convolutional layer produces 16 feature maps of size 24×24, which are subsequently flattened into a vector of size 1×9216 and passed to the fully connected part of the network. During error backpropagation, the error of the first fully connected layer neurons is distributed among all inputs and, consequently, divided by 9216. Additionally, in the error-correction process, the obtained error is further divided by 9216 to distribute it among all weights. This process leads to a significant reduction in the adjustment terms, thereby slowing down the overall learning speed.

To resolve this issue, we applied ALR (15) to the error-backpropagation process while setting the normalization factor in the error-correction process in (13) to 1. Figure 2 illustrates the learning rates for the first type of normalization compared to the default learning process. This approach was tested with and without ABS normalization. The results demonstrate that learning accuracy increases significantly from the beginning of the first epoch in both cases. However, while the model without ABS normalization experiences instability and divergence, the model with ABS normalization maintains stable learning throughout the training process.

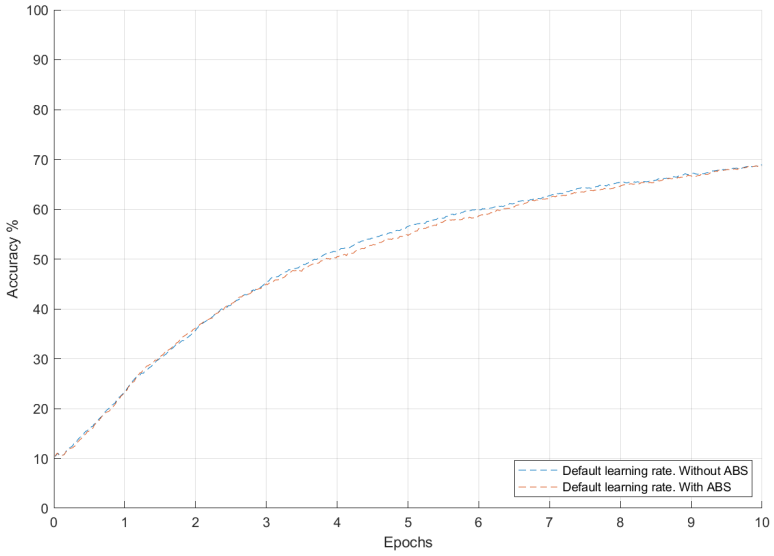

Figure 1. Accuracy of the model with default normalization factors and learning rates.

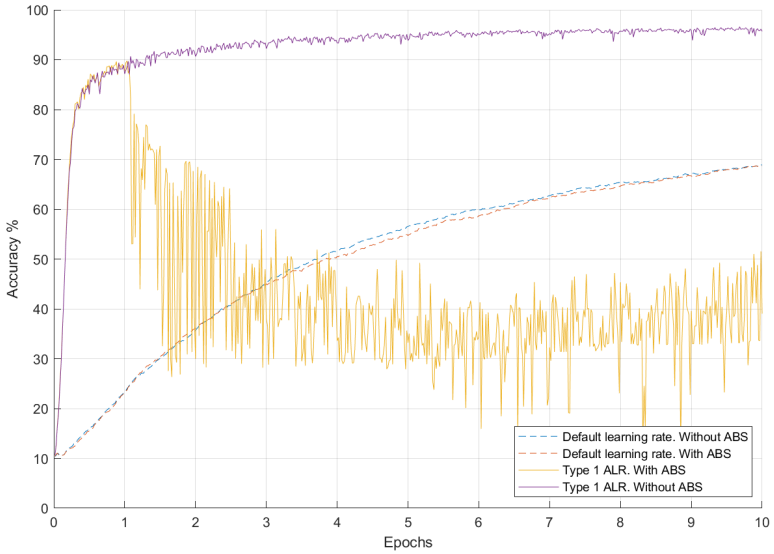

Figure 2. Accuracy of the model with the first ALR type.

Figure 3 presents the second ALR approach in comparison to the default learning process. This approach was also tested with and without ABS normalization. The results indicate that the second type of ALR also improves learning accuracy and accelerates model convergence. Furthermore, the figure illustrates different ALR rates and their respective impacts on model performance.

Based on the findings above, we were also interested in testing the model without normalization factors entirely. Two types of tests were conducted: one where the normalization factors in the first hidden layer were set to 1 and another where all normalization factors were set to 1. Both configurations were tested with and without ABS normalization. Figure 4 presents the accuracy rates of these models in comparison to the default learning process, providing further insights into the role of normalization in CNNMVN training.

It is important to highlight that, as observed in both ALR rules and the models without normalization, the application of ABS normalization played a crucial role in ensuring the convergence of the learning algorithm.

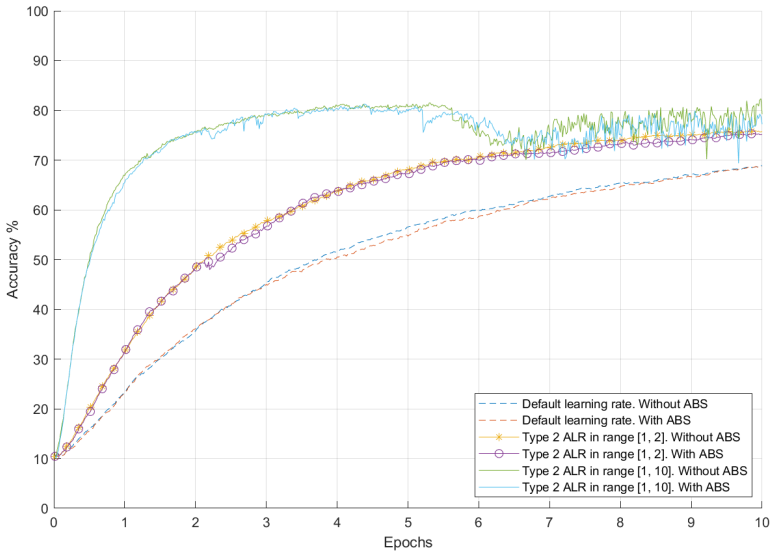

Figure 3. Accuracy of the model with the second ALR type.

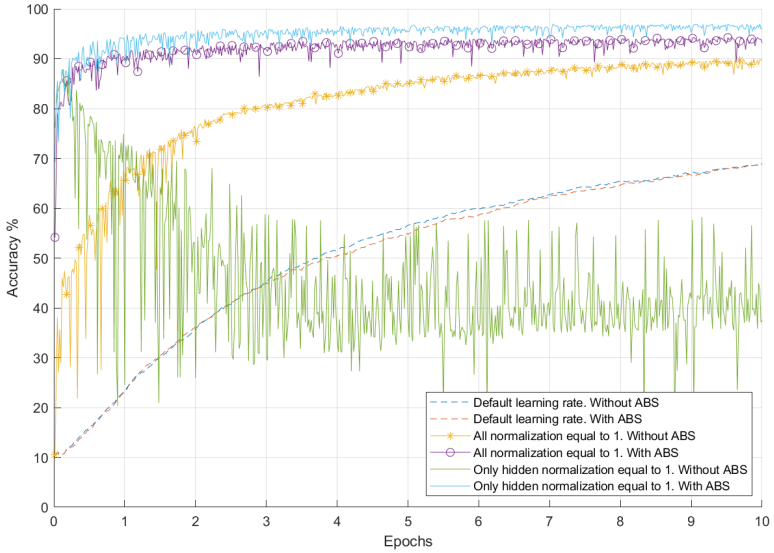

Figure 4. Accuracy of the model with only hidden fully-connected layer normalization set to 1 and model with all normalization factors set to 1.

6. Conclusion. In this paper we introduced adaptive learning rate (ALR) techniques for CNNMVN to improve training efficiency and convergence speed. Unlike traditional real-valued neural networks, CNNMVN follows an error-sharing principle instead of an optimization-based learning rule, allowing an unrestricted learning rate adjustments without the risk of overshooting local minima.

We developed two ALR strategies: one modifying learning rate coefficients in the error-correction formulas and another adjusting normalization factors in the error-backpropagation and error-correction formulas. Our experimental results on the MNIST dataset demonstrate that both approaches effectively accelerate learning and enhance model performance. Furthermore, our investigation of self-adaptive

normalization revealed that it plays a crucial role in stabilizing the learning process, particularly in preventing divergence when applying ALR techniques.

Overall, our findings highlight the potential of ALR in CNNMVN, offering a significant improvement in training efficiency while maintaining model stability. Future work may explore further refinements to ALR strategies, including adaptive adjustments tailored to specific network layers or tasks.

References

1. Aizenberg, I., & Vasko, A. (21–25 August, 2020). Convolutional Neural Network with Multi-Valued Neurons. In *2020 IEEE Third International Conference on Data Stream Mining & Processing (DSMP).* Lviv: Ukraine. https://doi.org/10.1109/DSMP47368.2020.9204076
2. Aizenberg, I., Herman, J., & Vasko, A. (26–29 October, 2022). A Convolutional Neural Network with Multi-Valued Neurons: A Modified Learning Algorithm and Analysis of Performance. In *2022 IEEE 13th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON).* New York: USA. https://doi.org/10.1109/UEMCON54665.2022.9965659
3. Aizenberg, I., & Vasko, A. (2024). Frequency-Domain and Spatial-Domain MLMVN-Based Convolutional Neural Networks. *Algorithms,* 17(8), 361. https://doi.org/10.3390/a17080361
4. Duchi, J., Hazan, E. & Singer, Y. (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research,* 12, 2121–2159. Retrieved from https://web.stanford.edu/~jduchi/projects/DuchiHaSi11.pdf
5. Hinton, G., Srivastava, N., & Swersky, K. (2012). Neural networks for machine learning. Lecture 6a Overview of mini-batch gradient descent. Retrieved from https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf
6. Kingma, D. P., & Ba, J. (2017). Adam: A Method for Stochastic Optimization. *Machine Learning,* arXiv. https://doi.org/10.48550/arXiv.1412.6980
7. Huang, J., Tang, T., Ding, Y., Tatikonda, S., Dvornek, N., Papademetris, X., & Duncan, J. S. (2020). AdaBelief Optimizer: Adapting Stepsizes by the Belief in Observed Gradients *Machine Learning,* arXiv. https://doi.org/10.48550/arXiv.2010.07468
8. Cheng, W., Pu, R., & Wang, B. (2025). AMC: Adaptive Learning Rate Adjustment Based on Model Complexity. *Mathematics,* 13(4), 650. https://doi.org/10.3390/math13040650
9. Takase, T., Oyama, S., & Kurihara, M. (2018). Effective neural network training with adaptive learning rate based on training loss. *Neural Networks: The Official Journal of the International Neural Network Society,* 101, 68–78. https://doi.org/10.1016/j.neunet.2018.01.016
10. Li, Q., Wang, B., Zhu, Y., Lioma, C., & Liu, Q. (2023). Adapting Pre-trained Language Models for Quantum Natural Language Processing. *Quantum Physics,* arXiv. https://doi.org/10.48550/arXiv.2302.13812
11. Zhao, W., & Huang, H. (2023). Adaptive orthogonal gradient descent algorithm for fully complex-valued neural networks. *Neurocomputing,* 546, 126358. https://doi.org/10.1016/j.neucom.2023.126358
12. Zhang, Y., & Huang, H. (2020). Adaptive complex-valued stepsize based fast learning of complex-valued neural networks. *Neural Networks,* 124, 233–242. https://doi.org/10.1016/j.neunet.2020.01.011
13. Dong, Z., & Huang, H. (2021). A training algorithm with selectable search direction for complex-valued feedforward neural networks. *Neural Networks,* 137, 75–84. https://doi.org/10.1016/j.neunet.2021.01.014
14. Aizenberg, I., & Moraga, C. (2007). Multilayer Feedforward Neural Network Based on Multi-valued Neurons (MLMVN) and a Backpropagation Learning Algorithm. *Soft Computing,* 11(2), 169–183. https://doi.org/10.1007/s00500-006-0075-5
15. Aizenberg, E., & Aizenberg, I. (09–12 December, 2014). Batch linear least squares-based learning algorithm for MLMVN with soft margins. In *2014 IEEE Symposium on Computational Intelligence and Data Mining (CIDM).* Orlando, FL: USA. https://doi.org/10.1109/CIDM.2014.7008147
16. Aizenberg, I. (2011). Complex-Valued Neural Networks with Multi-Valued Neurons. *Studies in Computational Intelligence.* (Vol. 353). Springer, Berlin Heidelberg.

https://doi.org/10.1007/978-3-642-20353-4
17. LeCun, Y., Cortes, C., & Burges, C. J. C. (August 9, 2024). The MNIST Database of handwritten digits. [Dataset]. Retrieved from https://www.kaggle.com/datasets/zalando-research/fashionmnist

Васько О. Ю., Брила А. Ю. Адаптивна нормалізація CNNMVN.

Адаптивні методи зміни швидкості навчання широко використовуються для підвищення ефективності навчання нейронних мереж, оскільки вони покращують швидкість та точність збіжності та зменшують ризик застрягання в локальних мінімумах або сідлоподібних точках. У цій статті ми представляємо підходи до адаптивної швидкості навчання для згорткової нейронної мережі з багатозначними нейронами (CNNMVN), яка є повністю комплекснозначною нейронною мережею, що оперує комплексними вхідними даними, комплексними вагами та комплекснозначними активаційними функціями.

На відміну від традиційних дійснозначних нейронних мереж, CNNMVN використовує принцип поділу помилки замість градієнтної оптимізації, що усуває проблему локальних мінімумів і дозволяє більш гнучко коригувати швидкість навчання. Ми пропонуємо дві стратегії адаптивної швидкості навчання (ALR), спеціально розроблені для CNNMVN. Перша стратегія модифікує коефіцієнти швидкості навчання у формулах корекції похибки, тоді як друга регулює параметри нормалізації у процесах зворотного поширення похибки та її корекції. Обидва методи динамічно адаптують швидкість навчання на основі точності на валідаційній вибірці.

Результати показують, що адаптивна швидкість навчання суттєво покращує швидкість збіжності та точність, особливо при поєднанні з самоналаштовуваною швидкістю навчання. Крім того, наше дослідження підкреслює вплив нормалізації на динаміку навчання та розглядає сценарії, у яких нормалізацію мінімізовано або повністю виключено.

Наші результати демонструють, що методи ALR покращують ефективність навчання CNNMVN, забезпечуючи надійну основу для оптимізації швидкості навчання в комплекснозначних нейронних мережах.

Ключові слова: згорткові нейромережі, комплекснозначні мережі, багатозначні нейрони, CNNMVN, MLMVN, розпізнавання зображень, частотна область.

Список використаної літератури

1. Aizenberg I., Vasko A. Convolutional Neural Network with Multi-Valued Neurons. *2020 IEEE Third International Conference on Data Stream Mining & Processing (DSMP).* Lviv : Ukraine, 21–25 August, 2020. pp. 72–77. DOI: https://doi.org/10.1109/DSMP47368.2020.9204076
2. Aizenberg I., Herman J., Vasko A. A Convolutional Neural Network with Multi-Valued Neurons: A Modified Learning Algorithm and Analysis of Performance. *2022 IEEE 13th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON).* New York : USA, 26–29 October, 2022. pp. 0585–0591. DOI: https://doi.org/10.1109/UEMCON54665.2022.9965659
3. Aizenberg I., Vasko A. Frequency-Domain and Spatial-Domain MLMVN-Based Convolutional Neural Networks. *Algorithms.* 2024. Vol. 17, No. 8. 361. DOI: https://doi.org/10.3390/a17080361
4. Duchi J., Hazan E., Singer Y. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research.* 2011. Vol. 12. pp. 2121–2159. URL: https://web.stanford.edu/j̃duchi/projects/DuchiHaSi11.pdf (date of access: 15.02.2025).
5. Hinton G., Srivastava N., and Swersky K. Neural networks for machine learning. Lecture 6a Overview of mini-batch gradient descent. 2012. URL: https://www.cs.toronto.edu/t̃ijmen/csc321/slides/lecture_slides_lec6.pdf (date of access: 16.02.2025).
6. Kingma D. P., and Ba J. Adam: A Method for Stochastic Optimization. *Machine Learning.* 2017. arXiv. DOI: https://doi.org/10.48550/arXiv.1412.6980

7. Huang J., Tang T., Ding Y., Tatikonda S., Dvornek N., Papademetris X., Duncan J. S. AdaBelief Optimizer: Adapting Stepsizes by the Belief in Observed Gradients *Machine Learning.* 2020. arXiv. DOI: https://doi.org/10.48550/arXiv.2010.07468
8. Cheng W., Pu R., and Wang B. AMC: Adaptive Learning Rate Adjustment Based on Model Complexity. *Mathematics.* 2025. Vol. 13, No. 4. 650. DOI: https://doi.org/10.3390/math13040650
9. Takase T., Oyama S., Kurihara M. Effective neural network training with adaptive learning rate based on training loss. *Neural Networks: The Official Journal of the International Neural Network Society.* 2018. Vol. 101. pp. 68–78. DOI: https://doi.org/10.1016/j.neunet.2018.01.016
10. Li Q., Wang B., Zhu Y., Lioma C., and Liu Q. Adapting Pre-trained Language Models for Quantum Natural Language Processing. *Quantum Physics.* 2023. arXiv. DOI: https://doi.org/10.48550/arXiv.2302.13812
11. Zhao W., Huang H. Adaptive orthogonal gradient descent algorithm for fully complex-valued neural networks. *Neurocomputing.* 2023. Vol. 546. 126358. DOI: https://doi.org/10.1016/j.neucom.2023.126358
12. Zhang Y., Huang H. Adaptive complex-valued stepsize based fast learning of complex-valued neural networks. *Neural Networks.* 2020. Vol. 124. pp. 233–242. DOI: https://doi.org/10.1016/j.neunet.2020.01.011
13. Dong Z., Huang H. A training algorithm with selectable search direction for complex-valued feedforward neural networks. *Neural Networks.* 2021. Vol. 137. pp. 75–84. DOI: https://doi.org/10.1016/j.neunet.2021.01.014
14. Aizenberg I., Moraga C. Multilayer Feedforward Neural Network Based on Multi-valued Neurons (MLMVN) and a Backpropagation Learning Algorithm. *Soft Computing.* 2007. Vol. 11, No. 2. pp. 169–183. DOI: https://doi.org/10.1007/s00500-006-0075-5
15. Aizenberg E., Aizenberg I. Batch linear least squares-based learning algorithm for MLMVN with soft margins. *2014 IEEE Symposium on Computational Intelligence and Data Mining (CIDM).* Orlando, FL: USA, 09–12 December, 2014. pp. 48–55. DOI: https://doi.org/10.1109/CIDM.2014.7008147
16. Aizenberg I. Complex-Valued Neural Networks with Multi-Valued Neurons. Studies in Computational Intelligence. Vol. 353. Springer, Berlin Heidelberg, 2011. DOI: https://doi.org/10.1007/978-3-642-20353-4
17. LeCun Y., Cortes C., Burges C. J. C. The MNIST Database of handwritten digits. August 9, 2024. [Dataset]. URL: https://www.kaggle.com/datasets/zalando-research/fashionmnist (date of access: 19.02.2025).

Recived 27.02.2025