Comparison of Deep Neural Network Learning Algorithms for Biomedical Image Processing

Oleh Berezsky[a)], Petro Liashchynskyi[a)], Oleh Pitsun[a)], Pavlo Liashchynskyi[a)] and Mykola Berezkyy [a)]

[a] *West Ukrainian National University, 11 Lvivska st., Ternopil, 46009, Ukraine*

Abstract

In recent years, the popularity of deep neural networks used for various problem-solving tasks has increased dramatically. The main tasks include image classification and synthesis using convolutional and generative-adversarial neural networks. These types of networks need large amounts of training data to achieve the required accuracy and performance. In addition, these networks have a long training time. The authors of the paper analyzed and compared the gradient-based neural network learning algorithms. The biomedical image classification with the use of a convolutional neural network of a given architecture was carried out. A comparison of learning algorithms (SGD, Adadelta, RMSProp, Adam, Adamax, Adagrad, and Nadam) was made according to the following parameters: training time, training loss, training accuracy, test loss, and test accuracy. For the experiments, the authors used the Python programming language, the Keras machine learning library, and the Google Colaboratory development environment, which provides free use of the Nvidia Tesla K80 graphics processor. For the experiments tracking and logging the authors used the Weights & Biases service.

Keywords

Machine learning, CNN, GAN, optimization algorithms, biomedical images.

1. Introduction

Neural networks are powerful tools used for solving a wide range of problems. A typical deep neural network consists of an input layer, several hidden layers, and an output layer. Any neural network optimizes a certain objective function depending on the type of problem. In recent years, the problems of image classification and synthesis with the use of convolutional and generative-adversarial neural networks have become relevant.

The use of neural networks to solve a specific problem involves solving the following tasks:

- selection of a training dataset;
- dataset preprocessing and augmentation if needed;
- selection of neural network architecture or designing it from scratch;
- selection of a learning algorithm;
- further architecture optimization and tuning.

Training a neural network means optimizing the parameters to achieve the minimum error value. Optimization of neural network parameters can be performed by various algorithms, which are called learning algorithms.

The purpose of this work is to compare neural network learning algorithms to classify biomedical images using a convolutional neural network.

IDDM-2022: 5th International Conference on Informatics & Data-Driven Medicine, November 18–20, 2022, Lyon, France
EMAIL: ob@wunu.edu.ua (A. 1); p.liashchynskyi@st.wunu.edu.ua (A. 2); o.pitsun@wunu.edu.ua (A. 3); pavloksmfcit@gmail.com (A. 4); mykolaberezkyy@gmail.com (A. 5);
ORCID: 0000-0001-9931-4154 (A. 1); 0000-0003-0646-7448 (A. 2); 0000-0003-0280-8786 (A. 3); 0000-0001-8371-1534 (A. 4) ; 0000-0001-6507-9117 (A. 5);

2. Literature review

Learning algorithms are divided into first-order or second-order algorithms and evolutionary algorithms. First-order algorithms are based on the calculation of the first derivative of the error function. Therefore, these algorithms are also called gradient algorithms. Second-order algorithms use the second derivative to select the direction of error minimization. Evolutionary algorithms are built on the basis of genetic algorithms.

In [1], the author analyzed the known gradient learning methods and provided their visualization.

The authors of article [2] compared three evolutionary algorithms using a hybrid neural network in forecasting downstream river flow based on areal precipitation.

In the article [3], the authors compared several gradient optimization methods for a simple convolutional neural network. The Nadam algorithm showed the best results.

In the article [4], the author described the implementation of neural networks in the FPGA environment. This implementation allows for speeding up the learning processes of neural networks due to the use of parallel processing. As a learning algorithm, the author used a simple gradient descent.

In the research study [5], the author substantiated the relevance of improving neural network training methods for object classification and segmentation problems. The author developed a method that reduces the training time of neural networks based on nonlinear dynamics. The improved method is based on the gradient descent method with delayed feedback.

In these publications, researchers mostly paid attention to the analysis of existing algorithms. Only some of the authors compared learning algorithms.

Therefore, the limitations of these publications are that they only partially address the problem of learning algorithms comparison. Most of the publications are just about learning algorithms review when solving a bigger problem.

The main goal of any learning algorithm is to minimize the learning error and optimize the network parameters. Modern classifiers [6, 7] require large amounts of training data to achieve high accuracy. In the work [8], the authors described the process of biomedical image classification and synthesis using convolutional and generative-adversarial neural networks. The process of training these networks is time-consuming. The training time can be reduced with an adequate selection of the learning algorithm.

Therefore, the actual task is the comparison of learning algorithms for biomedical image classification.

3. Analysis of learning algorithms

Modern algorithms for learning neural networks are based on error backpropagation and the gradient descent method. These algorithms are called gradient or first-order algorithms.

An important parameter of algorithms is the learning rate. This parameter controls how far to move in the direction opposite to the gradient of the function in one step. If the learning rate is low, the training time of the neural network can increase significantly. If the learning rate is high, the neural network may not reach the minimum error value [9]. Formally, it can be presented as follows:

$$\theta = \theta - \alpha \nabla J(\theta), \tag{1}$$

where θ refers to neural network parameters,
ɑ is a learning rate,
$\nabla J(\theta)$ is a gradient of the optimization function (loss).

The disadvantage of gradient descent is that the network parameters can be updated only after passing the full training dataset.

Among other gradient learning algorithms, stochastic gradient descent and mini-batch gradient descent are distinguished.

Stochastic gradient descent (SGD) differs from the usual one in that the network parameters are updated after each training iteration [10]. Therefore, when using this learning algorithm, the parameters of the neural network are updated much more often.

Mini-batch gradient descent uses data packets to update parameters [11]. The training dataset is divided into packets of the same size. Then each of the packets is sent to the network input, the gradient is calculated and the parameters are updated. Equation (1) can be represented in the following way:

$$\theta = \theta - \alpha \nabla J(\theta;\ B(i)), \tag{2}$$

where $B(i)$ is a package of training examples.

Let us analyze the variations of gradient descent methods.

Adagrad. The essence of this algorithm is that the learning rate adapts according to the network parameters [12]. The algorithm sets a lower learning rate for parameters that are associated with frequent features in the dataset. Then the equation with iterations will have the following form:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{G_t + \epsilon}} \odot g_t, \tag{3}$$

where G is a diagonal matrix, where each of the diagonal elements is the sum of the squares of the gradients with respect to the parameters θ at all previous iterations, including t,
ϵ is a parameter with a small value that prevents division by 0 (usually $1e - 10$),
g is a gradient of the optimization function, $\nabla J(\theta)$.

The advantage of this algorithm is that a researcher does not need to set the learning rate manually. The authors use the default value for the learning rate, which is 1.0 [12].

The disadvantage of the algorithm is an accumulation of gradients from previous iterations. This leads to a decrease in the learning rate and a minor update of the network parameters.

Adadelta. This algorithm is an improved version of the previous algorithm. The Adadelta algorithm reduces the size of the matrix of accumulated gradients to a particular fixed value [13].

$$\Delta\theta_t = -\frac{RMS[\Delta\theta]_{t-1}}{RMS[g]_t} g_t, \tag{4}$$

$$\theta_{t+1} = \theta_t + \Delta\theta_t, \tag{5}$$

where RMS is a root mean square value,
g is a gradient of the optimization function, $\nabla J(\theta)$.

The advantage of this algorithm is in no need for setting the initial value of the learning speed.

RMSProp. This algorithm is similar to the Adadelta algorithm. It was developed by Geoffrey Hinton [14]. The equations that describe the operation of the algorithm are as follows:

$$E_t = 0.9E[g^2]_{t-1} + 0.1g_t^2, \tag{6}$$

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{E_t + \epsilon}} g_t. \tag{7}$$

RMSprop was developed at around the same time as Adadelta. These algorithms solve the problem of monotonically decreasing learning rates in the Adagrad algorithm.

Adam та Adamax. Unlike the two previous algorithms, in addition to the squares of the previous g^2 gradients, the Adam algorithm also stores the previous gradients g:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \tag{9}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2, \tag{10}$$

where m, v are estimates of the mean and variance of the gradients, respectively [15].

The rule for updating parameters in this algorithm is as follows:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t, \tag{11}$$

where $\hat{v}_t = \frac{v_t}{1-\beta_2^t}, \hat{m}_t = \frac{m_t}{1-\beta_1^t}$.

The authors suggest the following values for the parameters: $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1e - 8$. The Adamax is a variant of the Adam algorithm.

Nadam. Nadam is a combination of RMSProp and Momentum algorithms. The first algorithm accumulates the squares of the gradient values, and the second algorithm accumulates the values of the previous gradients.

4. Dataset and augmentation

A training set of cytological images with a size of 64x64 pixels was used for the experiments. The initial dataset contains about 100 images. Therefore, the dataset was expanded to approximately 800 images using affine distortions. The Python programming language and the Rudi library [14] were used to expand the training data set.

Cytological images form a subset of biomedical images. Cytological images are images of cells of the organism. Examples of cytological images are shown in Figure 1.

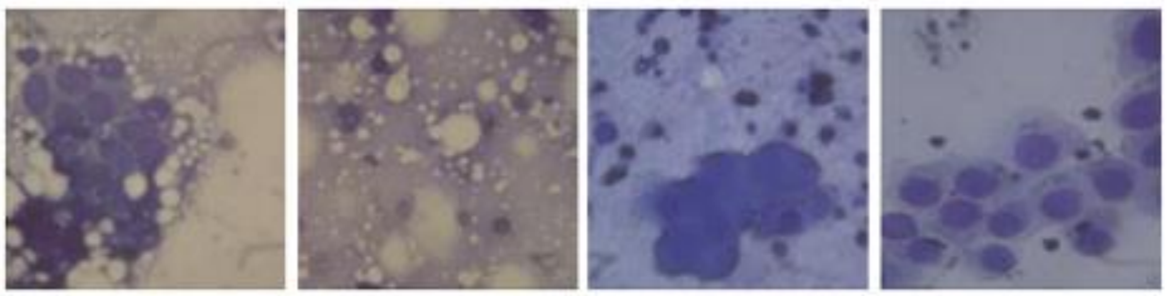

Figure 1. Cytological images

Cytological image processing and analysis are reflected in works [16-18].

5. CNN architecture design

To compare the gradient descent-based training methods, a convolutional neural network model was built. As an input, the network accepts color cytological images with a size of 64x64 pixels and outputs a class label. The sequence of layers is given in Table 1.

Table 1
Model summary

Layer	Output shape	Layer param
Input	64x64x3	
Conv	32x32x64	kernel size = 5
Batch Norm	32x32x64	
Leaky Relu	32x32x64	slope = 0.2
Conv	16x16x128	kernel size = 5
Batch Norm	16x16x128	
Leaky Relu	16x16x128	slope = 0.2
Max Pool	8x8x128	
Dropout	8x8x128	rate = 0.5
Conv	4x4x256	kernel size = 3
Batch Norm	4x4x256	
Leaky Relu	4x4x256	slope = 0.2
Conv	2x2x512	kernel size = 3
Batch Norm	2x2x512	
Leaky Relu	2x2x512	slope = 0.2
Dropout	2x2x512	rate = 0.5
Flatten	2048	
Dense	4	
Softmax	4	

As can be seen from Table 1, the network consists of several repeating blocks. Each block consists of a sequence of convolution layers, batch normalization, an activation layer, and a dropout layer. Each convolutional layer reduces by half the input volume.

The model is compiled using the categorical cross-entropy loss function. The number of learning epochs is 30.

The dataset is divided into learning and testing in the ratio of 80% to 20%. The batch size is set to 64.

The Tensorflow 2 library and the Python programming language were used to build the model and conduct experiments. The experiments were conducted in the Google Colaboratory environment on an Nvidia Tesla K80 graphics processor.

6. Experiments

The results of the optimization of neural network parameters based on gradient descent are shown in Table 2.

Table 2
Optimizers

Name	Learning rate
Adam	0.001
SGD	0.01
RMSprop	0.001
Nadam	0.001
Adadelta	0.001
Adagrad	0.001
Adamax	0.001

All optimizer parameters are set to the default values specified in the Tensorflow library. The comparison of learning algorithms was made on the basis of the network training time, the value of the loss function, and the classification accuracy on the test dataset. The results of the experiments are shown in Table 3 and in the figures below.

Table 3
Experimental results

Name	Training time (m)	Loss	Test loss	Accuracy	Test accuracy
Adam	4.10	0.05	0.77	0.9797	0.7625
SGD	2.50	0.07	1.29	0.9731	0.7462
RMSprop	3.39	0.08	0.45	0.9719	0.895
Nadam	3.55	0.04	2.79	0.9822	0.66
Adadelta	4.35	0.64	0.45	0.7319	0.8425
Adagrad	3.49	0.13	**0.12**	0.9491	**0.9488**
Adamax	4.45	0.06	0.30	0.9775	0.9162

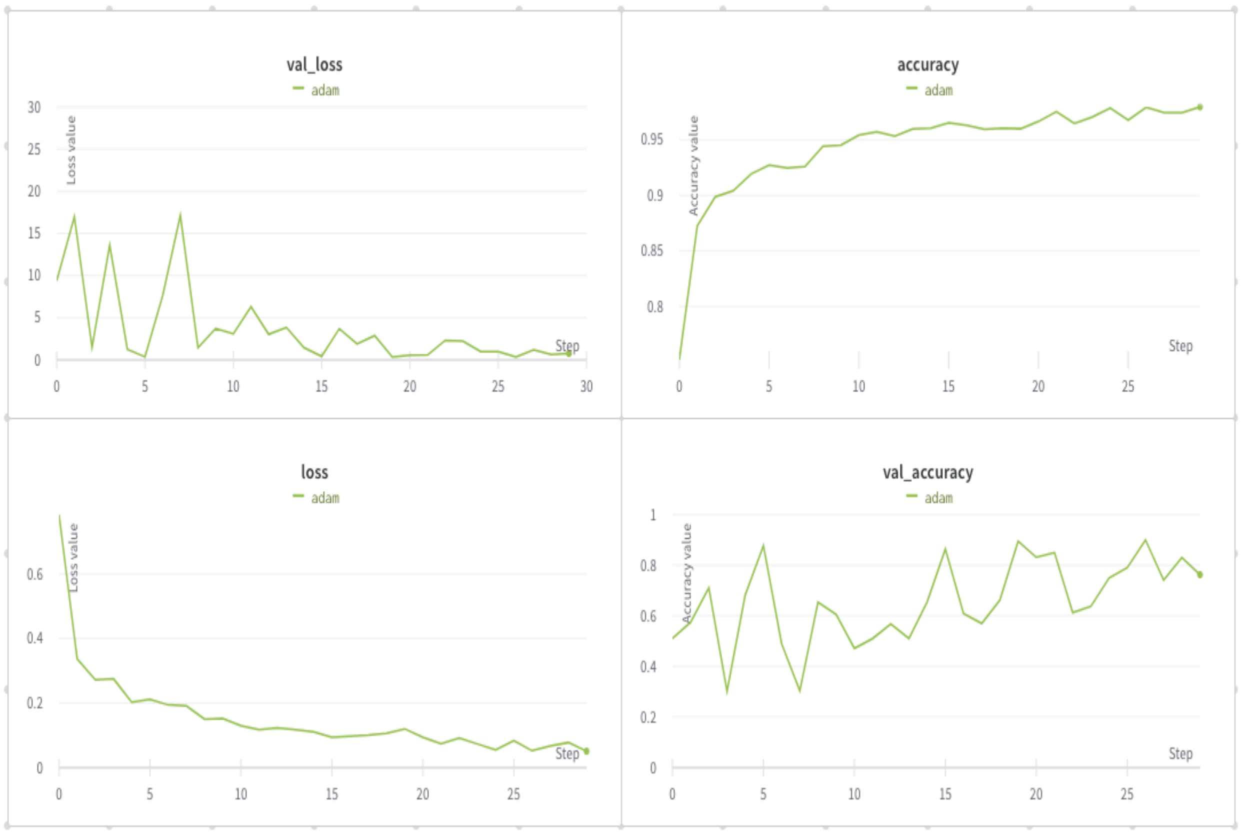

Figure 2: Model with Adam optimizer

Figure 2 shows the error graphs and accuracy graphs on the training and test datasets for the Adam optimizer. The accuracy curves on the training and test datasets show that the network is being retrained. This happens because there is a significant difference in accuracy between the training and test datasets. Accuracy on the training dataset was ~98%, and accuracy on the test dataset was

~76%.

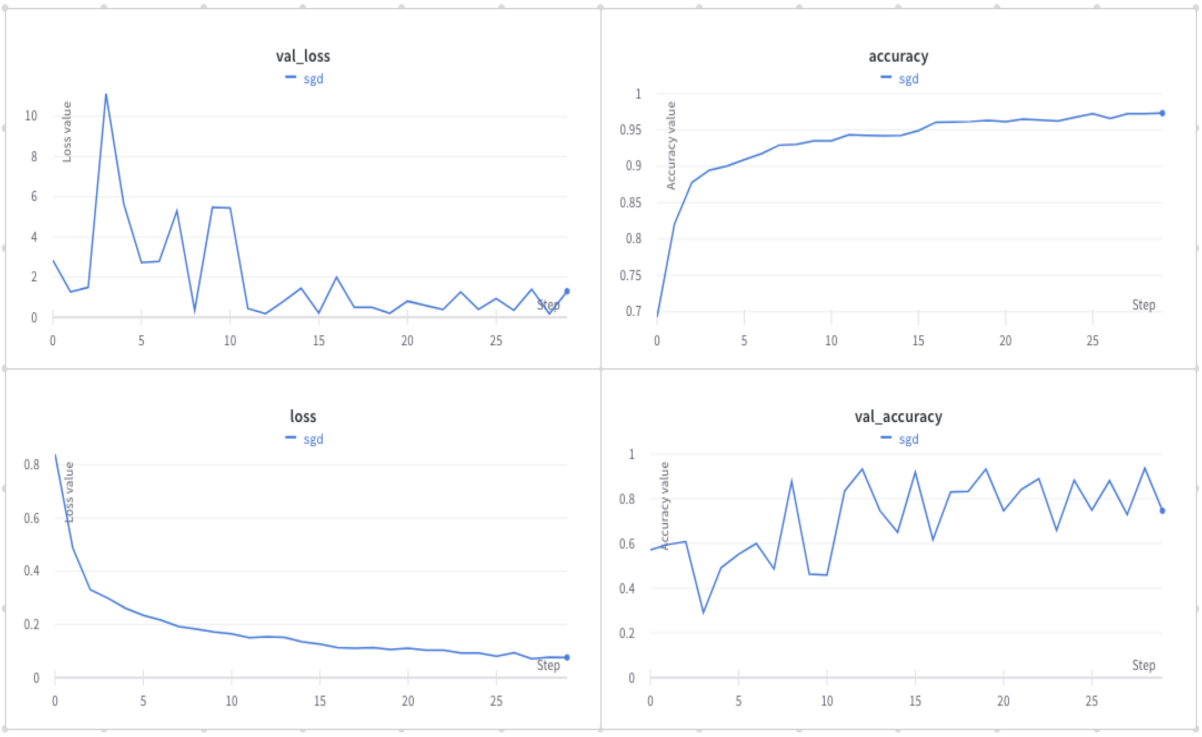

Figure 3: Model with SGD optimizer

Figure 3 also shows a significant difference between the accuracy values on the training and test datasets: ~97% and ~75%, respectively.

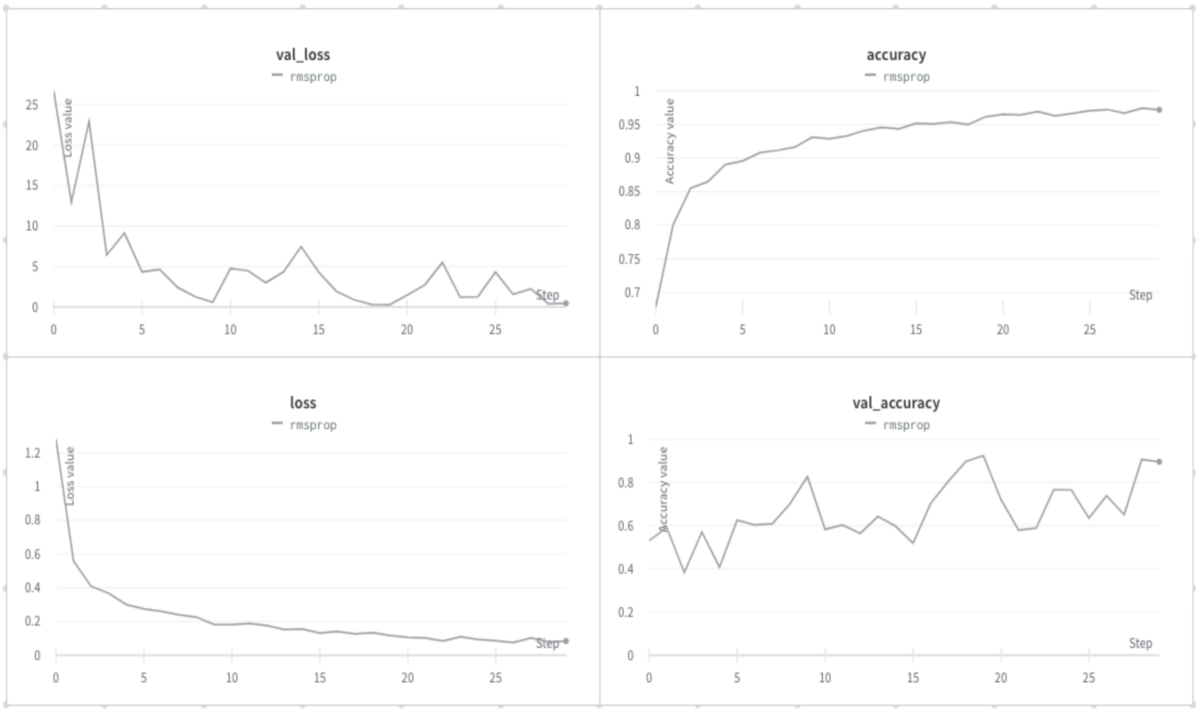

Figure 4: Model with RMSprop optimizer

The classification accuracy for the model with the RMSprop optimizer on the training and test datasets was ~97% and ~89%, respectively (fig.4).

The model with the Nadam optimizer also demonstrates significant overtraining. Accuracy on the training and test datasets is ~98% and 66%, respectively (fig.5).

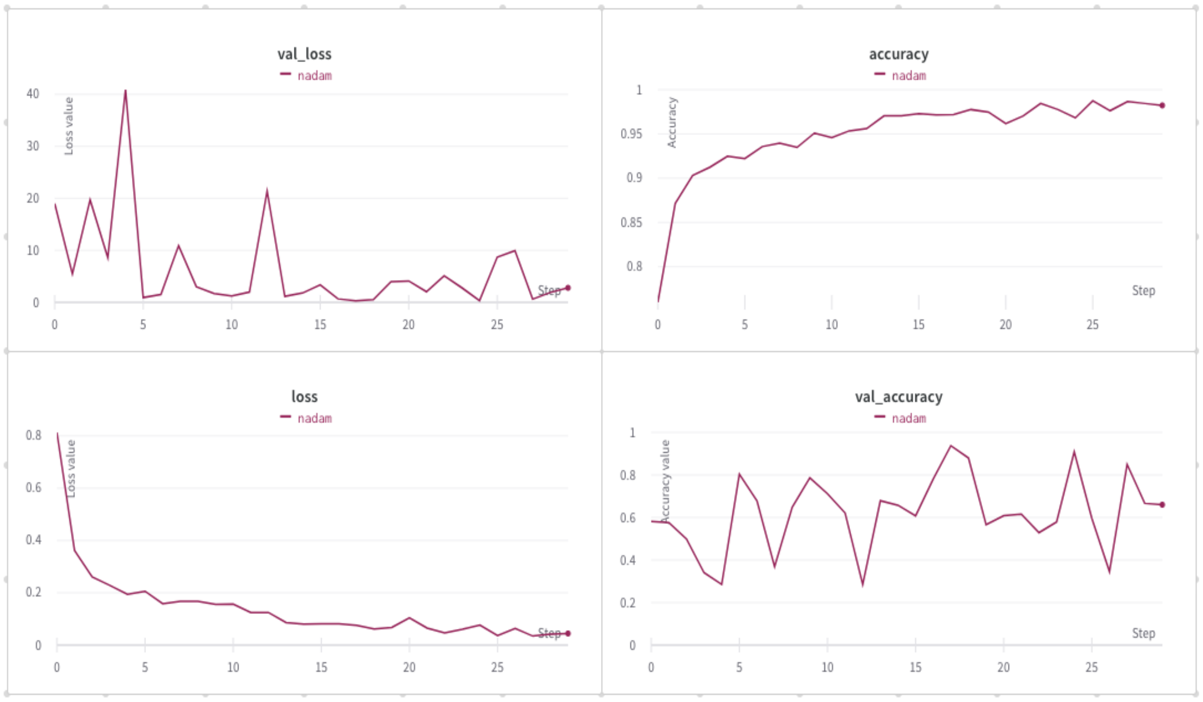

Figure 5: Model with Nadam optimizer

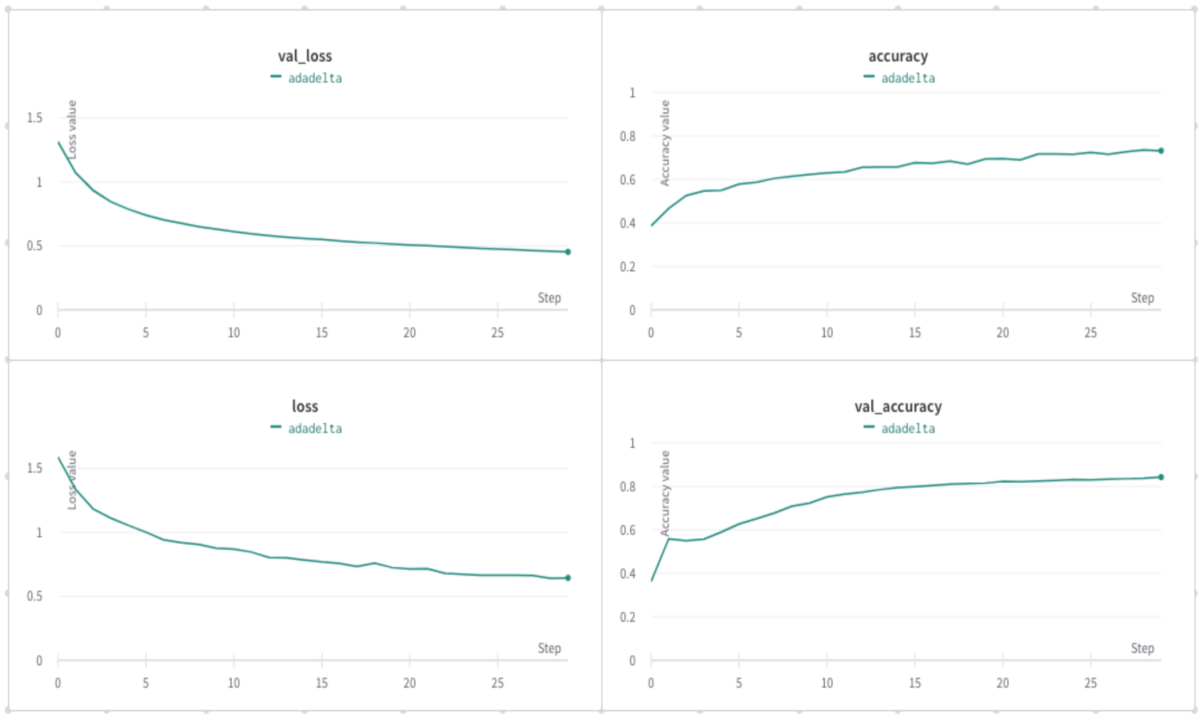

Figure 6: Model with Adadelta optimizer

As can be seen from Figure 6, the error and accuracy graphs are quite smooth. However, the classification accuracy on the training dataset was only ~73%. This is explained by another problem, namely underfitting. There are several options for its solution, such as an increase in the number of training epochs or an increase in the complexity of the used model.

The accuracy values on the training and test data sets are almost the same for the model with the Adagrad optimizer and equal to ~95% (fig.7).

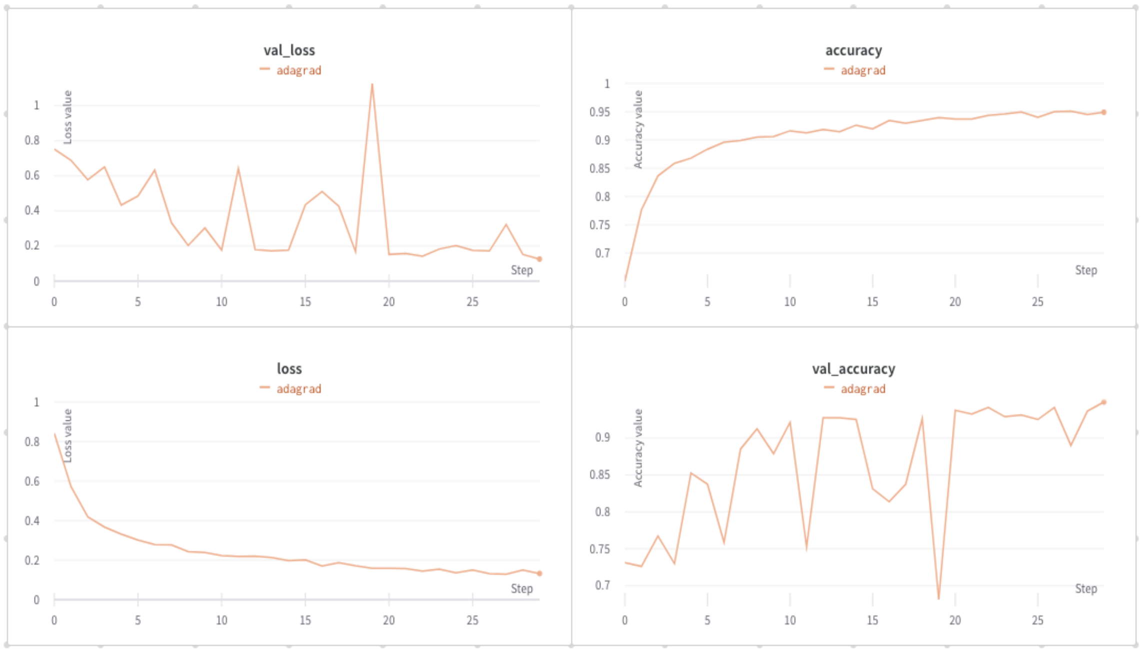

Figure 7: Model with Adagrad optimizer

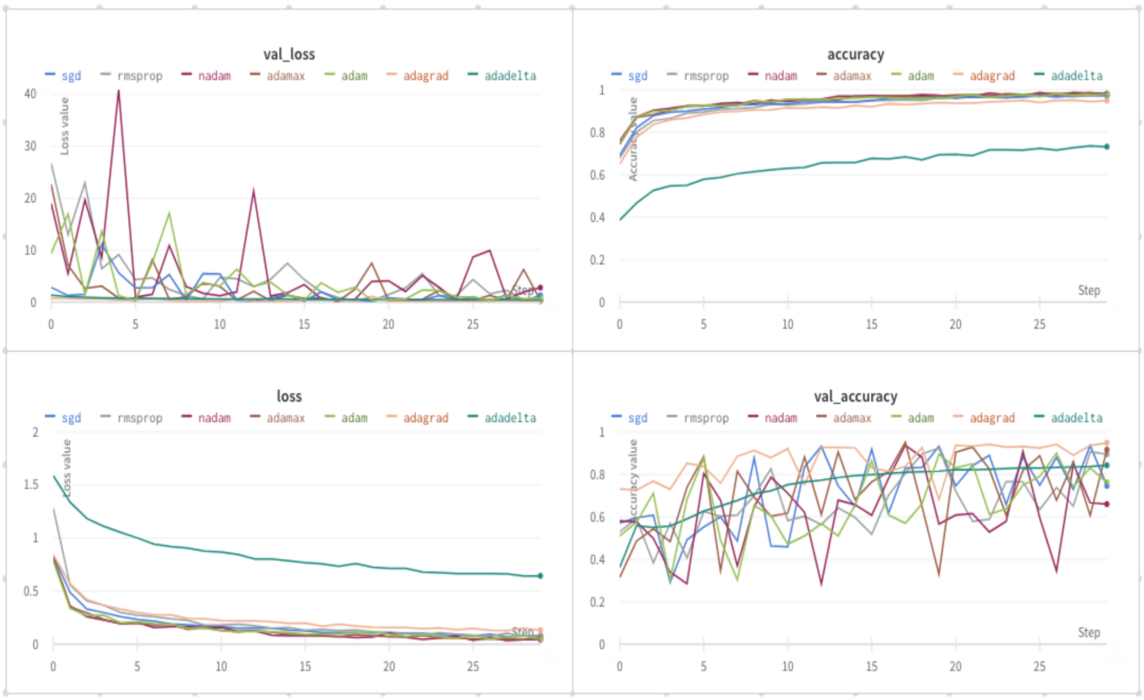

Figure 8: All optimizers visualization

Figure 8 shows that the error and accuracy curves on the test dataset for almost all models are not smooth, so the networks are retrained. This is evidenced by the significant difference in accuracy values on the training and test datasets. This problem can be solved by simplifying the neural network model or by increasing the number of images in the training dataset. On the other hand, the model with the Adadelta optimizer has an inverse problem called underfitting. To solve this problem, several techniques can be applied, such as increasing the complexity of the model or increasing the number of training epochs.

7. Conclusions

The results of the study are as follows:

1. The authors of the research study conducted a comparative analysis of the gradient descent-based algorithms for optimizing neural network parameters (Adam, SGD, RMSprop, Nadam, Adadelta, Adagrad, and Adamax). The comparison was made according to the criteria of network training time, loss function values, and classification accuracy on the cytological image dataset.
2. Based on the cytological image dataset and the developed convolutional neural network model, the four best optimizers were selected according to the val_accuracy parameter: Adamax, Adadelta, Adagrad, and RMSprop.
3. The graphs of val_loss and val_accuracy on the test data set for optimizers (except Adadelta) are not smooth. Unlike other algorithms, Adadelta is an optimization algorithm with an adaptive learning rate. Therefore, parameters are updated with a smaller step, and during the training, there is no problem with retraining. As a result, the accuracy and error curves on the training and test datasets almost coincide and are smooth.
4. Since Adadelta is an adaptive algorithm, it is necessary to use a higher training rate at the beginning. This will significantly reduce the training time and ensure the convergence of the model.

In this work, only one neural network model was used. In future research, it is planned to apply discussed optimizers to more complex and huge models. The further research direction will also cover the application of optimizers for generative-adversarial networks.

8. References

[1] Ruder, Sebastian. "An overview of gradient descent optimization algorithms." arXiv preprint arXiv:1609.04747 (2016). URL: https://arxiv.org/abs/1609.04747

[2] X.Y. Chen, K.W. Chau, A.O. Busari. A comparative study of population-based optimization algorithms for downstream river flow forecasting by a hybrid neural network model, Engineering Applications of Artificial Intelligence, Volume 46, Part A, 2015, Pages 258-268, ISSN 0952-1976, https://doi.org/10.1016/j.engappai.2015.09.010.

[3] E. M. Dogo, O. J. Afolabi, N. I. Nwulu, B. Twala and C. O. Aigbavboa, "A Comparative Analysis of Gradient Descent-Based Optimization Algorithms on Convolutional Neural Networks," 2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS), 2018, pp. 92-99, https://ieeexplore.ieee.org/document/8769211/

[4] Khuzhakhmetova, A. Sh, A. V. Semenyutina, and V. A. Semenyutina. "Deep neural network elements and their implementation in models of protective forest stands with the participation of shrubs." International journal of advanced trends in computer science and engineering 9.4 (2020): 6742-6746.

[5] Smorodin A. V. Methods of learning neural networks based on nonlinear dynamics: diss. dr. Philos. Sciences: 122 / Smorodin Andriy Vyacheslavovich – Odesa, 2022. – 169 p.

[16] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[7] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. https://doi.org/10.1109/CVPR.2016.90

[8] Berezsky, O., Pitsun, O., Liashchynskyi, P., Derysh, B., Batryn, N. (2023). Computational Intelligence in Medicine. In: Babichev, S., Lytvynenko, V. (eds) Lecture Notes in Data Engineering, Computational Intelligence, and Decision Making. ISDMCI 2022. Lecture Notes on Data Engineering and Communications Technologies, vol 149. Springer, Cham. https://doi.org/10.1007/978-3-031-16203-9_28

[9] Amari, Shun-ichi. "Backpropagation and stochastic gradient descent method." Neurocomputing 5.4-5 (1993): 185-196. https://doi.org/10.1016/0925-2312(93)90006-O

[10] Ketkar, Nikhil. "Stochastic gradient descent." Deep learning with Python. Apress, Berkeley, CA, 2017. 113-132.

[11] Hinton, Geoffrey, Nitish Srivastava, and Kevin Swersky. "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent." Cited on 14.8 (2012): 2.

[12] 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 http://jmlr.org/papers/v12/duchi11a.html

[13] Zeiler, M. D. (2012). ADADELTA: An Adaptive Learning Rate Method. URL: http://arxiv.org/abs/1212.5701

[14] Hinton, G., Srivastava, N., Swersky, K. Overview of mini-batch gradient descent. URL: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

[15] Kingma, D. P., & Ba, J. L. (2015). Adam: a Method for Stochastic Optimization. International Conference on Learning Representations, 1–13. https://arxiv.org/abs/1412.6980

[16] Berezsky, O., Pitsun, O., T. Dolynyuk, T., Dubchak, L., Savka, N., Melnyk, G. & Teslyuk, V. “Cytological Image Classification Using Data Reduction”. II International Workshop Informatics & Data-Driven Medicine (IDDM 2019). Lviv, Ukraine. November 11-13, 2019. http://ceur-ws.org/Vol-2488/paper2.pdf.

[17] Berezsky, O., Pitsun, O., Datsko, T., Derysh, B., Tsmots, I. & Tesluk, V. “Specified diagnosis of breast cancer on the basis of immunogistochemical images analysis”, IDDM’2020: 3rd International Conference on Informatics & Data-Driven Medicine, November 19–21, 2020, Växjö, Sweden. pp. 129-135. http://ceur-ws.org/Vol-2753/short5.pdf,

[18] Berezsky, O., Pitsun, O., Dubchak, L., Berezka, K., Dolynyuk, T. & Derish, B. Cytological Images Clustering. In: Shakhovska N., Medykovskyy M.O. (eds) Advances in Intelligent Systems and Computing V. CSIT 2020. Advances in Intelligent Systems and Computing. 2021; vol 1293. Springer, Cham. https://doi.org/10.1007/978-3-030-63270-0_12.