Pruning connections in a fully connected neural network allows to remove redundancy in the structure of the neural network and thus reduce the computational complexity of its implementation while maintaining the resulting characteristics of the classification of images entering its input. However, the issues of choosing the parameters of the pruning procedure have not been sufficiently studied at the moment. The choice essentially depends on the configuration of the neural network. However, in any neural network configuration there is one or more multilayer perceptrons. For them, it is possible to develop universal recommendations for choosing the parameters of the pruning procedure. One of the most promising methods for practical implementation is considered – the iterative pruning method, which uses preprocessing of input signals to regularize the learning process of a neural network. For a specific configuration of a multilayer perceptron and the MNIST (Modified National Institute of Standards and Technology) dataset, a database of handwritten digit samples proposed by the US National Institute of Standards and Technology as a standard when comparing image recognition methods, dependences of the classification accuracy of handwritten digits and learning rate were obtained on the learning step, pruning interval, and the number of links removed at each pruning iteration. It is shown that the best set of parameters of the learning procedure with pruning provides an increase in the quality of classification by about 1 %, compared with the worst set in the studied range. The convex nature of these dependencies allows a constructive approach to finding a neural network configuration that provides the highest classification accuracy with the minimum amount of computational costs during implementation

Keywords: multilayer perceptron, neural network, pruning, learning curve, weight coefficients, image classification

UDC 681.3.07: 004.8
DOI: 10.15587/1729-4061.2022.253103

DETERMINATION OF THE INFLUENCE OF THE CHOICE OF THE PRUNING PROCEDURE PARAMETERS ON THE LEARNING QUALITY OF A MULTILAYER PERCEPTRON

Oleg Galchonkov
Corresponding author
PhD, Associate Professor*
E-mail: o.n.galchenkov@gmail.com
Alexander Nevrev
PhD, Associate Professor*
Bohdan Shevchuk
Postgraduate Student*
Nikolay Baranov
Senior Lecturer*
*Department of Information Systems
Institute of Computer Systems
National University Odessa Polytechnic
Shevchenko ave., 1, Odessa, Ukraine, 65044

Received date 13.01.2022
Accepted date 24.02.2022
Published date 28.02.2022

How to Cite: *Galchonkov, O., Nevrev, A., Shevchuk, B., Baranov, N. (2022). Definition of the influence of the choice of the pruning procedure parameters on the quality of training of a multilayer perceptron. Eastern-European Journal of Enterprise Technologies, 1 (9 (115)), 75–83. doi: https://doi.org/10.15587/1729-4061.2022.253103*

1. Introduction

The use of deep neural networks is becoming more and more widespread in various practical applications, in particular, in image classification problems [1]. Along with the development of convolutional neural networks, a large number of neural network architectures have appeared that provide the same classification quality as convolutional neural networks, but require less computation. These are networks such as MLP-Mixer (multilayer perceptron mixer) [2], Vision Transformer (ViT) [3], Compact Transformers [4], ConvMixer (Transformer using convolutions for mixing) [5], External Attention Transformer (Transformer with external attention) [6], FNet (Transformer using Fourier transform) [7], gMLP (MLPs with gating – multilayer perceptrons with element-wise multiplication) [8], Swin Transformer (Transformer with shifted windows) [9] and similar ones. Despite the variety of architectures, all of them have a multilayer perceptron (MLP) at the output, and besides this, inside the architecture.

Numerous studies have shown that fully connected neural networks for many practical problems have a large redundancy, which can be eliminated without loss of the resulting quality [10] or even with some improvement [11]. One of the most popular approaches to reduce redundancy is to thin out the connections in the original fully connected neural network. A fully connected network with a large number of connections, providing the specified characteristics, is used as the initial neural network. The purpose of link pruning is to reduce computational costs while maintaining the resulting characteristics. For example, in the popular AlexNet and VGG-16 neural networks (a deep convolutional neural network developed in 2014 by the Oxford University Computer Vision Group (Visual Geometry Group) and researchers from Google DeepMind) it is possible to reduce the number of connections by 9 and 13 times respectively without worsening the resulting characteristics [12]. Since the appearance of the first works on reducing redundancy in neural networks [13], a large number of various approaches have been developed [11]. However, they do not cover the whole variety of architectures of modern neural networks. In addition, new practical applications are constantly emerging that require an increase in the quality of image classification, while limiting the used computing power. Therefore, research in this direction is relevant.

2. Literature review and problem statement

It is possible to distinguish the following approaches to reducing the computational complexity of neural networks by eliminating redundancy and while maintaining the resulting output characteristics.

Knowledge Distillation – involves the use of special objective functions for learning a simple model based on the features of the structure and set of weights of a pre-learned fully connected parent model. In [14], a new type of ensemble classifier is proposed, which contains a main neural network and a number of small specialized neural networks. The latter are designed to classify those images on which the main network is wrong. In [15], it was proposed to divide the studied architectures of neural networks into blocks and study their influence on the efficiency of the final architecture. In [16], the analysis of the inner layers of the neural network is used to remove neurons with a low contribution to the resulting classification accuracy. However, building the structure of a simple model that can be effectively learned under the control of the parent neural network is a rather difficult task for practical implementation [17].

Methods for lowering the rank of the weight matrix (Low-rank Decomposition) use the reduction of weight coefficients corresponding to small eigenvalues of the original coefficient matrix. In [18], an algorithm for approximating convolutional filters in neural networks was proposed. In [19], it was proposed to simultaneously use the methods of compression and low-rank decomposition to the entire neural network. However, the problem of finding eigenvalues and vectors of high-dimensional matrices is very computationally complex, which can lead to a significant deterioration in the resulting characteristics of the resulting structures of neural networks.

Parameter Quantization provides a reduction in computational complexity by replacing floating-point calculations with fixed-point calculations with a reduced bit depth of representation of weight coefficients. In [20], a three-stage pipeline is proposed, which includes neuron pruning, iterative reduction in the number of digits in the weight representation, and Huffman coding. This allows to significantly reduce the memory requirements of the computing device. The use of dynamically changing bit depth in [21] made it possible to obtain high performance on the CIFAR-10 (Canadian Institute for Advanced Research) and MNIST datasets. However, implementations of such approaches are most suitable for implementation on programmable logic arrays, and not on universal computers, and with a significant decrease in the capacity, they can lead to significant losses in the resulting characteristics.

Network Pruning involves the removal of either single connections or entire structures in a neural network based on the calculation of an additional function of the importance of weight coefficients. This direction seems to be the most promising from the point of view of practical implementation on multi-core and multi-processor computers due to the good parallelizability of calculations. In earlier works, the matrix of second derivatives of the loss function was used as such a function [13]. However, the calculation of such a matrix for large networks requires a very large amount of computation at each iteration, which initiated the development of algorithms with a less complex importance function [11]. In particular, structural methods have been developed for convolutional neural networks that involve the removal of entire channels [17]. These methods are well suited for the convolutional layers of these networks, but do not cover the multilayer perceptron at the output of convolutional neural networks, since there are no explicitly allocated channels. For pruning fully connected layers in convolutional neural networks and simply multilayer perceptrons, the simplest way is to iteratively remove the smallest weight coefficients [12]. In this case, the periods of learning the neural network and deleting connections alternate. The presence of learning intervals allows the learning algorithm to reduce the values of the weight coefficients for those connections that are least important in the current network configuration. Obviously, with this approach, the balance between the duration of learning intervals and the number of links removed after it is extremely important. Too many connections removed at one time, or too early removal, when the neural network has not yet had time to sufficiently learn the structure of its connections, can lead to degradation of the neural network. On the other hand, too rare removal of a small number of connections delays the learning process of the neural network. However, in [12] it is not described how the parameters of the pruning procedure were chosen and their influence on the resulting characteristics of the neural network was not investigated. Another factor that has a significant impact on the resulting quality of learning of the neural network and the elimination of redundancy is the possibility of overlearning the neural network. Here, overlearning of a neural network is understood as a situation when, due to the presence of redundant neurons and connections, the network is learned too well on learning data, which leads to worse results on test data. The standard approach to deal with this phenomenon is regularization [22].

In [23], a practical implementation of regularization was proposed for learning neural networks using the L_0 norm. It has been shown that it can speed up learning. However, since the L_0 weight norm is not differentiable, it cannot be directly included in the objective function. This significantly complicates the practical implementation of this form of regularization. In [24], channel pruning in neural convolutional networks is considered. The L_1 and L_2 norms are used for regularization. This allows efficient pruning of channels between convolutional layers without overlearning. However, the regularization method considered in [24] for channel pruning is not suitable for link pruning in the output multilayer perceptron, since there are no explicit channels there. In [25], the stabilization of the characteristics of the neural network during learning with pruning is carried out using Dropout. Dropout is a simple and efficient way to regularize. The principle of operation of this procedure is to disconnect from learning the connections selected at each iteration by a random number generator. However, disabling connections from learning lead to some slowdown in convergence. From the point of view of regularization, it seems promising to use pre-distortions of input signals (Augmentation). In [26], the influence of image preprocessing on the characteristics of convolutional neural networks LeNet, Network3 (a neural network implemented using the network3.py library) and DropConnect (a convolutional neural network using DropConnect regularization is an improved version of Dropout) was analyzed. An improvement in the resulting characteristics of neural networks is shown. However, the links were not prunned out in [26]. In [27], a study was made of the potential characteristics of a multilayer perceptron in classifying images from MNIST. When learning perceptrons, preprocessing of input images was used. In [28], the best practices for document processing are considered, one of them is the expansion of the learning data set due to image preprocessing. Both in [27] and in [28], the pruning of links was not considered. It was shown in [29] that the simultaneous use of link pruning in MLP and input signal pre-distortion makes it possible to obtain not only the regularization of the learning process, but also higher resulting characteristics. However, in this work, the main focus is on the study of the effectiveness of the joint use of pruning and pre-distortion of input signals. The issues of the influence of the parameters of the pruning procedure on the resulting characteristics were not considered. Therefore, it is of interest to study the influence of the choice of parameters of the link pruning procedure in MLP on the learning rate and the resulting characteristics.

Thus, the analysis of the literature shows the presence of a large number of approaches to the implementation of the learning procedure with pruning and ensuring its stability due to regularization. At the same time, there are no studies on the effect of pruning procedure parameters on the resulting characteristics. The most promising from the point of view of research with the subsequent construction of a technique for choosing the parameters of the pruning procedure is an iterative pruning procedure with regularization due to data preprocessing. It is obvious that this choice essentially depends on the dimension of the neural network and the signals coming to its input. However, the study of the influence of the choice of parameters of the pruning procedure for a specific neural network structure and input data set will allow to navigate when working with other structures and data sets.

3. The aim and objectives of the study

The aim of the study is to determine the dependence of the resulting quality of the classification of a multilayer perceptron on such parameters of the learning algorithm as the learning step, the pruning period and the number of links to be removed at a time. This will make it possible to ensure the maximum quality of image classification in specific practical applications with the chosen architecture of the neural network.

To achieve the aim, the following objectives were set:

– for a specific data set (MNIST[30]) and neural network architecture, perform pruning learning with pruning procedure parameters in a wide range of values, obtain typical learning curves, and plot the resulting classification quality versus these parameters;

– to formulate a methodology for finding the best parameters of the pruning procedure, depending on the type of learning curve and the dynamics of changes in the resulting classification quality.

4. Materials and methods of research

4. 1. Description of the neural network and its learning algorithm

For comparability of the results, it was used the same neural network as in [29]. It contains an input layer of 784 nodes, a first hidden layer of 256 neurons, a second hidden layer of 128 neurons, and an output layer of 10 neurons. The activation function for all neurons is sigmoid. The input of the neural network is preprocessed images of handwritten digits from the MNIST set [30], each of which contains 784 pixels. The response of the neural network is determined by the number of the output at which the maximum signal. Constant step stochastic gradient descent (SGD) is used for learning. Since the sigmoid is used as the activation function, Javier's [31] initialization was used for initialization. In the initial state of the neural network, all connections between layers are present, the weight coefficients are random numbers with a normal distribution, zero mean, and a variance inversely proportional to the number of connections included in the corresponding neuron. The formation of neural network output signals and the learning algorithm are described by the following equations [29]:

– the input signal of the first hidden layer:

$$X_{in-h1} = W_{in-h1} \times X_{in}, \tag{1}$$

where in – vector of neural network input signals, dimension:

n_{in}=784,

W_{in-h1} – matrix of weight coefficients between the input layer and the first hidden layer, dimension $n_{h1}\times n_{in}$; n_{h1}=56 – the number of neurons in the first hidden layer, output signal of the first hidden layer:

$$X_{o-h1} = f_{act}\left(X_{in-h1}\right), \tag{2}$$

where $f_{act}\left(x\right)=1/\left(1+e^{-x}\right)$ – activation function, second hidden layer input:

$$X_{in-h2} = W_{h1-h2} \times X_{o-h1}, \tag{3}$$

where W_{h1-h2} – matrix of weight coefficients between the first and second hidden layers, dimension $n_{h2}\times n_{h1}$; n_{h2}=128 – the number of neurons in the second hidden layer, the output signal of the second hidden layer:

$$X_{o-h2} = f_{act}\left(X_{in-h2}\right), \tag{4}$$

the input and output signals of the output layer respectively:

$$X_{in-o} = W_{h2-o} \times X_{o-h2}, \tag{5}$$

$$X_{out} = f_{act}\left(X_{in-o}\right), \tag{6}$$

where W_{h2-o} – matrix of weight coefficients between the second hidden layer and the output layer, dimension $n_o\times n_{h2}$; n_o=10 – the number of neurons in the output layer (number of neural network outputs), error vector at the output of the neural network (dimension n_o):

$$E_{out} = X_{tar} - X_{out}, \tag{7}$$

where X_{tar} – vector of known correct output signals of the neural network, dimension, n_o, contains all zeros, except for 1 at the place that matches the number shown in the input figure, the error vector at the output of the second hidden layer (dimension n_{h2}):

$$E_{h2} = W_{h2-o}^{T} \times E_{out}, \tag{8}$$

error vector at the output of the first hidden layer (dimension n_{h1}):

$$E_{h1} = W_{h1-h2}^{T} \times E_{h2}, \tag{9}$$

weight update equations:

$$W_{h2-o} = W_{h2-o} + \mu\left(\left(E_{out} \cdot X_{out} \cdot \left(1 - X_{out}\right)\right) \times X_{o-h2}^{T}\right), \tag{10}$$

$$W_{h1-h2} = W_{h1-h2} + \mu\left(\left(E_{h2} \cdot X_{o-h2} \cdot \left(1 - X_{o-h2}\right)\right) \times X_{o-h1}^{T}\right), \tag{11}$$

$$W_{in-h1} = W_{in-h1} + \mu\left(\left(E_{h1} \cdot X_{o-h1} \cdot \left(1 - X_{o-h1}\right)\right) \times X_{in}^{T}\right), \tag{12}$$

where the operation «·» denotes element-wise multiplication, μ – learning step, a scalar value.

Pruning connections in the neural network is carried out using auxiliary $H_{h2\text{-}o}$, $H_{h1\text{-}h2}$, $H_{in\text{-}h1}$ with the corresponding dimensions $n_o \times n_{h2}$, $n_{h2} \times n_{h1}$, $n_{h1} \times n_{in}$:

$$W_{h2-o} = W_{h2-o} \cdot H_{h2-o}, \tag{13}$$

$$W_{h1-h2} = W_{h1-h2} \cdot H_{h1-h2}\ , \tag{14}$$

$$W_{in-h1} = W_{in-h1} \cdot H_{in-h1}\ , \tag{15}$$

where the elements of the matrices $H_{h2\text{-}o}$, $H_{h1\text{-}h2}$, $H_{in\text{-}h1}$ have unit initial values, when any connection is thinned out, the corresponding unit is replaced by zero.

Every L learning epochs, k percent of non-zero elements of the matrices, $H_{h2\text{-}o}$, $H_{h1\text{-}h2}$, $H_{in\text{-}h1}$, corresponding to the elements of the weight coefficient matrices $W_{h2\text{-}o}$, $W_{h1\text{-}h2}$, $W_{in\text{-}h1}$ with the smallest modules, are reset to zero.

One epoch – learning using 60,000 images.

4. 2. Description of the MNIST dataset

The MNIST dataset [30] contains 60,000 images of handwritten digits for neural network learning and 10,000 for testing. Each image has a dimension of 28×28 pixels and is accompanied by information about what is shown on it. Based on this information, the X_{tar} vectors used in equation (7) are generated. The color of each pixel is encoded as an eight-digit binary integer in the range from 0 to 255. For pre-processing, these values are converted to floating point format, and for further use in equations (1)...(12) they are normalized to the range from 0 to 1. Examples of numbers from the MNIST set are presented in Fig. 1.

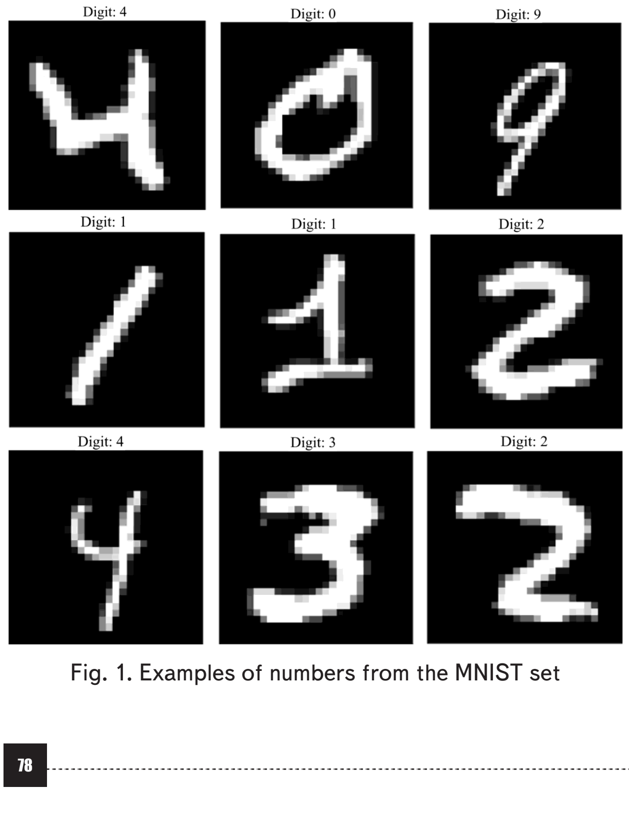

Fig. 1. Examples of numbers from the MNIST set

This is one of the most common sets of images on which various configurations and algorithms for learning neural networks are tested.

4. 3. Data preprocessing

When learning a neural network, each of the images received at its input is subjected to preliminary processing, which consists in sequentially performing the following operations:

– rotation of the image relative to the center by a random angle in the range from –15 to +15 degrees;

– horizontal shift by a random value in the range from –0.05 to +0.05 of the image size in width;

– vertical shift by a random value in the range from –0.05 to +0.05 from the image size in height;

– expansion/compression of the image relative to the center by a factor in the range from 0.95 to 1.05;

– increase in contrast – if the color value of a pixel was less than or equal to 100, then it was assigned the value 0, if the color value was more than 100, then it was assigned the value 255.

For each of the operations, a separate random number generator with a uniform distribution was used. When performing each of the operations, the free pixels were filled with the same color as the nearby filled pixels.

Such preprocessing ensures the uniqueness of the images received at the input of the neural network and the regularization of the learning process [29]. The task of studying the influence of preprocessing parameters on the resulting characteristics of the neural network was not set in this work. Studies of this influence were carried out in [26–28].

Test images from the MNIST set are not involved in learning and are not preprocessed.

5. Experimental study of the dependence of the resulting MLP characteristics from the parameters of the pruning procedure

5. 1. Results of learning a multilayer perceptron for different parameters of the pruning procedure

The research program was written in Python using the Num.py, Tensorflow, and Keras libraries. Graphs were built using the Matplotlib library. The program implemented equations (1)–(15) exactly. Input images were preprocessed using functions from the Keras library. The program was run on the cloud platform Colab [32]. The Python interpreter and used libraries are preinstalled on this platform. Likewise, popular datasets are loaded on the platform, including MNIST.

Fig. 2 shows neural network learning curves with different learning steps for pruning intervals L equal to 1, 2, and 3 epochs, and the number of links removed k equal to 1, 2, 3, and 4 percent. The learning of weight coefficients was carried out for each image supplied to the input. After every 1000 learning iterations, the quality of classification by the neural network of images from the learning set (60 thousand images) was calculated at the current values of the matrix elements $W_{h2\text{-}o}$, $W_{h1\text{-}h2}$, $W_{in\text{-}h1}$. Classification quality refers to the ratio of correctly classified images to the total number of images. Points on the curves in Fig. 2 are obtained by averaging 60 classification quality values calculated during learning over the corresponding one epoch. Fig. 3 shows the image classification quality graphs from the test set, corresponding to the learning curves in Fig. 2. That is, the points of the graphs in fig. 3 were obtained with the same matrices of weight coefficients that were obtained during learning at the corresponding learning moments. The difference is that the classification quality was calculated for images from the test set (10,000 images).

The dependences of the learning curves from the number of removed links for the same learning step and pruning interval $L=1$ epoch are shown for the learning data in Fig. 4 and for the test data in Fig. 5.

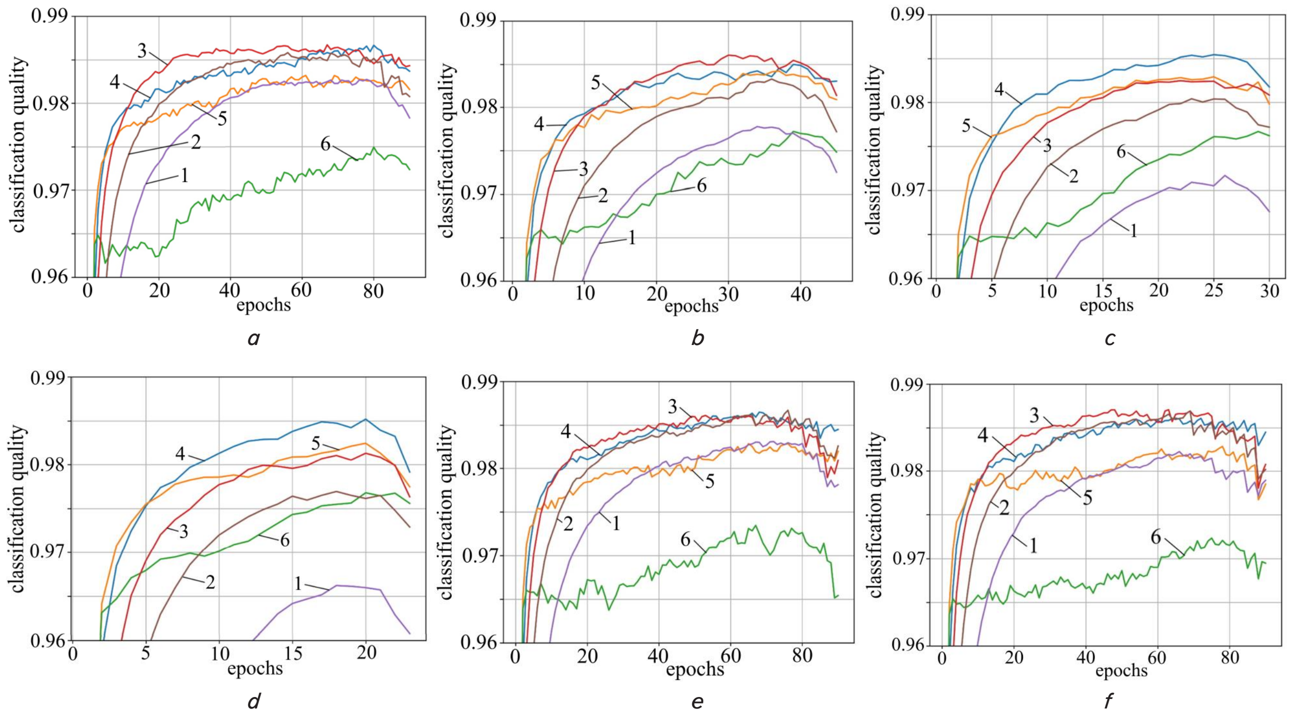

Fig. 2. Learning curves of a multilayer perceptron for various values of the learning step and parameters of the pruning procedure: *a* — $L=1$, $k=1$; *b* — $L=1$, $k=2$; *c* — $L=1$, $k=3$; *d* — $L=1$, $k=4$; *e* — $L=2$, $k=2$; *f* — $L=3$, $k=3$; 1 — $\mu=0.0012$; 2 — $\mu=0.0025$; 3 — $\mu=0.005$; 4 — $\mu=0.01$; 5 — $\mu=0.02$; 6 — $\mu=0.04$

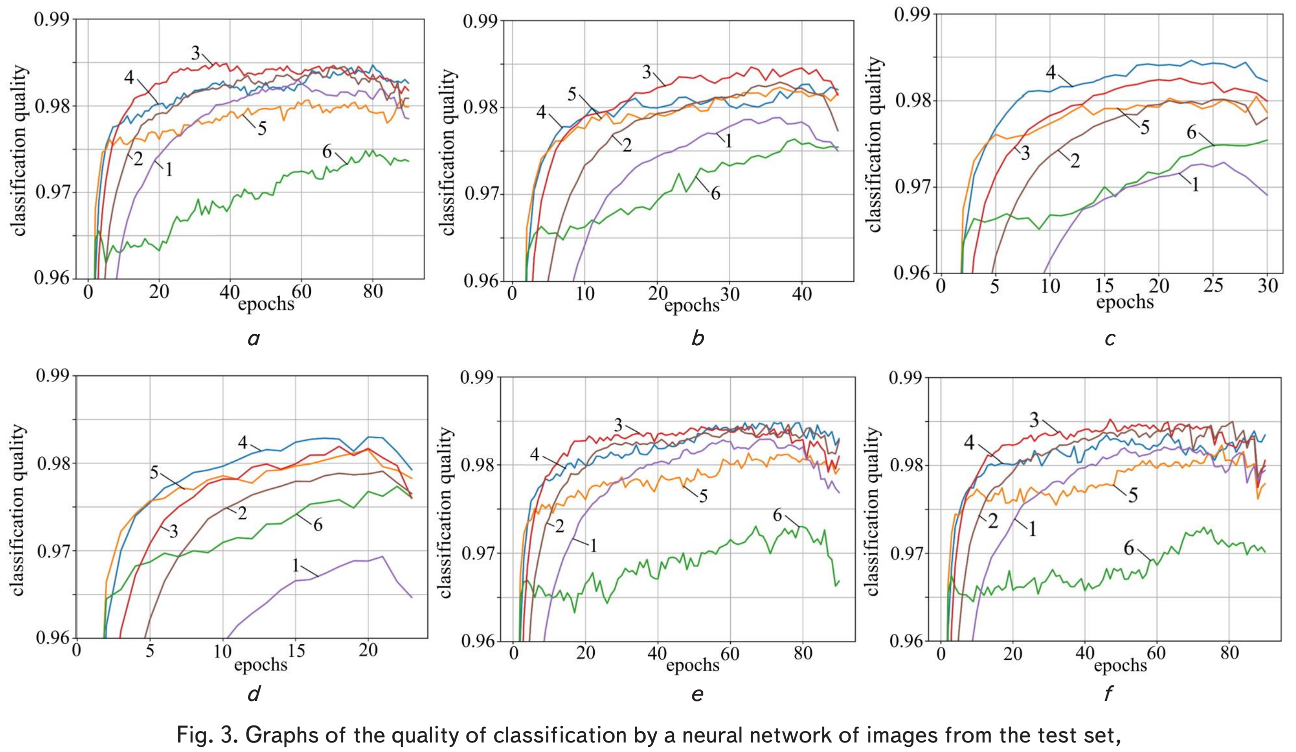

Fig. 3. Graphs of the quality of classification by a neural network of images from the test set, corresponding to the learning curves shown in Fig. 2: *a* — $L=1$, $k=1$; *b* — $L=1$, $k=2$; *c* — $L=1$, $k=3$; *d* — $L=1$, $k=4$; *e* — $L=2$, $k=2$; *f* — $L=3$, $k=3$; 1 — $\mu=0.0012$; 2 — $\mu=0.0025$; 3 — $\mu=0.005$; 4 — $\mu=0.01$; 5 — $\mu=0.02$; 6 — $\mu=0.04$

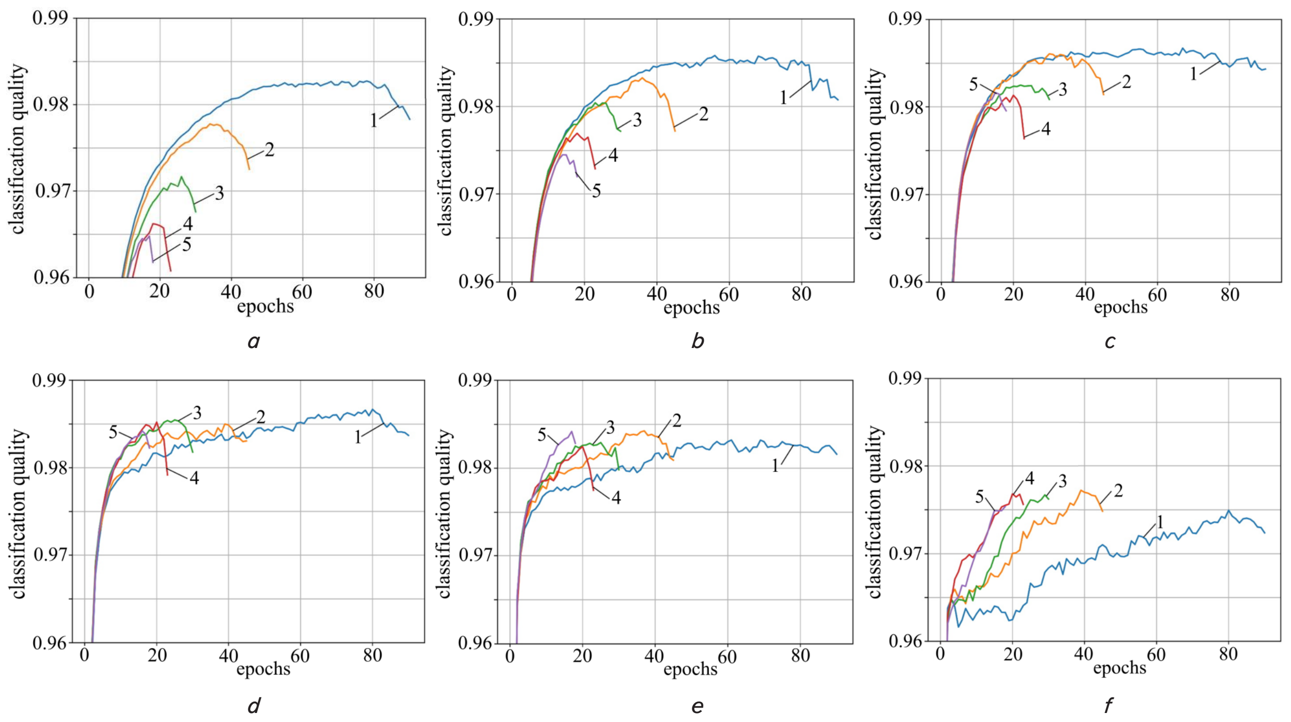

Fig. 4. Dependence of the learning curve from the number of connections to be removed according to the learning data with $L=1$ and the same learning step: *a* – μ=0.0012; *b* – μ=0.0025; *c* – μ=0.005; *d* – μ=0.01; *e* – μ=0.02; *f* – μ=0.04; 1 – $k=1$; 2 – $k=2$; 3 – $k=3$; 4 – $k=4$; 5 – $k=5$

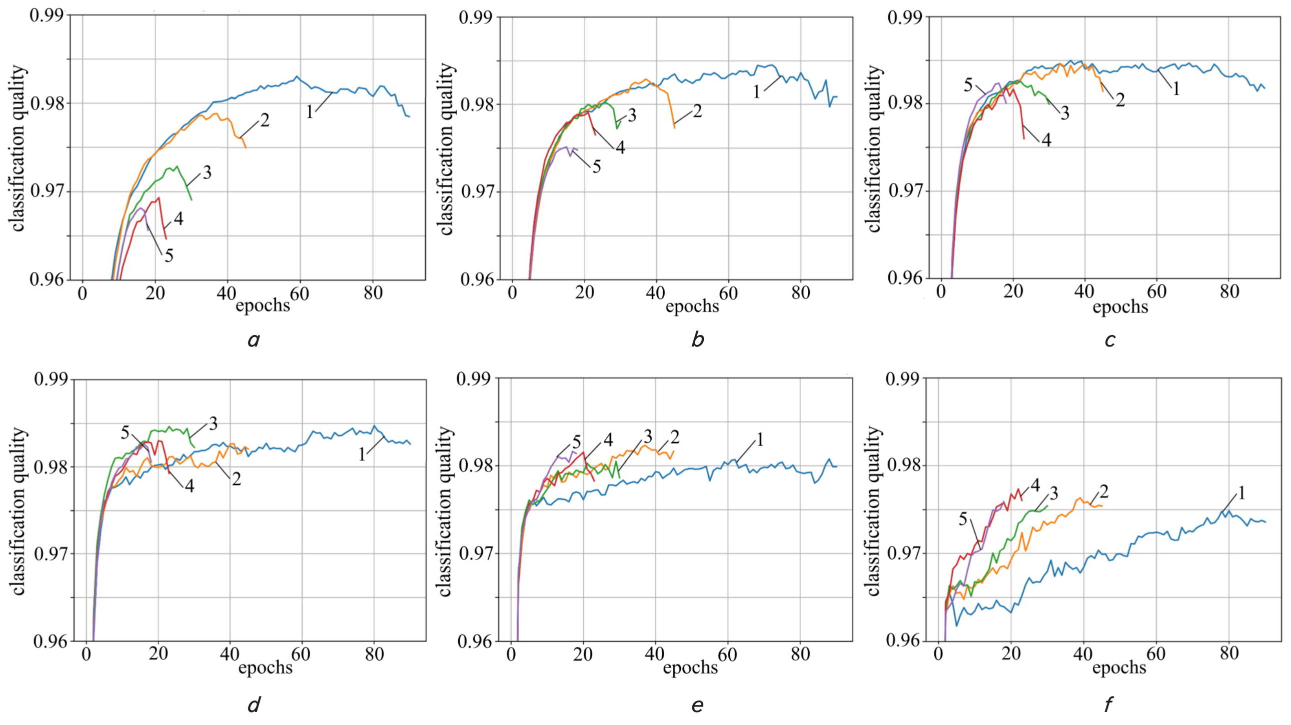

Fig. 5. Dependence of the quality of classification from the number of links removed according to the data for testing at $L=1$ and the same learning step: *a* – μ=0.0012; *b* – μ=0.0025; *c* – μ=0.005; *d* – μ=0.01; *e* – μ=0.02; *f* – μ=0.04; 1 – $k=1$; 2 – $k=2$; 3 – $k=3$; 4 – $k=4$; 5 – $k=5$

The graphs in Fig. 2–5 show the following:

– for the chosen configuration of the neural network and the input data set used, the difference in the resulting characteristics of the neural network when choosing the worst and the best set of parameters for the learning procedure is about 1 %,

– for the chosen configuration of the neural network and the used set of input data, the redundancy is completely removed when about 80 % of the connections are excluded. This can be seen from the fact that, regardless of the learning step and the parameters of the pruning procedure, further removal of links leads to a deterioration in the classification;

– 1 % pruning after each learning epoch provides a slightly better result than 2 % pruning every two epochs and 3 % pruning every three epochs;

– when pruning links after each epoch, an increase in the percentage of links removed leads to a deterioration in the quality of the classification;

– there is the best value of the learning step that provides the greatest value of correct classification. With a small learning step, the neural network does not have time to learn, and with a large learning step, degradation of the learning process occurs. The corresponding dependences of the maximum achieved classification accuracy depending from the learning step are shown in Fig. 6, and the same range of learning steps ensures the fastest achievement of these maxima – Fig. 7.

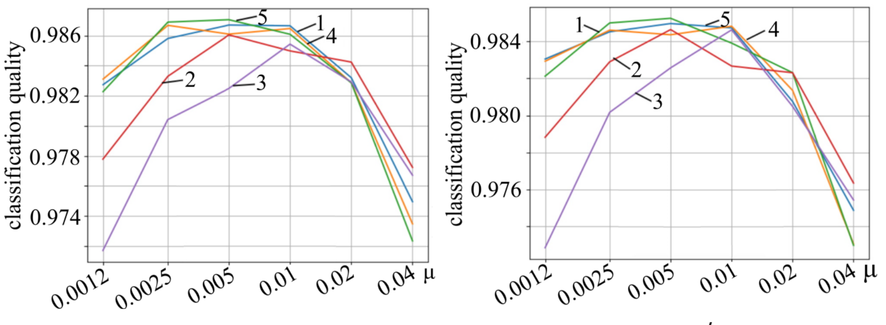

Fig. 6. Dependence of the maximum classification accuracy from the learning step and parameters of the pruning procedure: *a* — for images from the learning set; *b* — for images from the test set; 1 — *L*=1, *k*=1; 2 — *L*=1, *k*=2; 3 — *L*=1, *k*=3; 4 — *L*=2, *k*=2, 5 — *L*=3, *k*=3

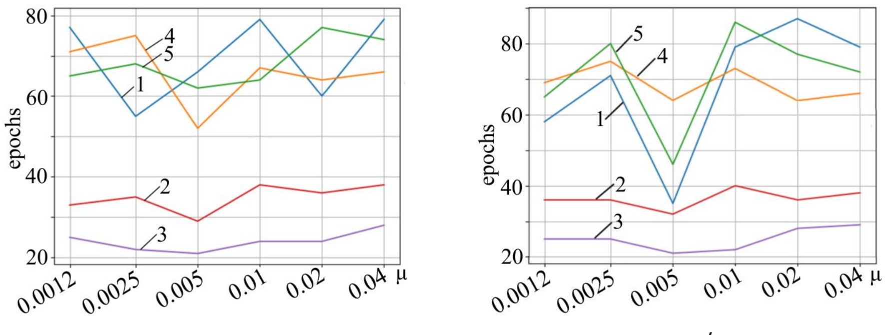

Fig. 7. Dependence of the number of epochs to achieve the maximum classification accuracy from the learning step and parameters of the pruning procedure: *a* — for images from the learning set; *b* — for images from the test set; 1 — *L*=1, *k*=1; 2 — *L*=1, *k*=2; 3 — *L*=1, *k*=3; 4 — *L*=2, *k*=2, 5 — *L*=3, *k*=3

The graphs in Fig. 6 show that the difference in the maximum achieved classification accuracy for different parameters of the pruning procedure within the studied range of parameters can reach 1 %. Moreover, the data in Fig. 7 show that the learning step that provides the maximum classification accuracy simultaneously provides the minimum number of iterations to achieve this maximum. It should be noted that when the maximum accuracy is reached, the learning process of the neural network does not stop, since the second goal pursued is the removal of all redundancy from the network architecture.

5. 2. Technique for finding the best parameters of the pruning procedure

The technique involves multiple learning of the neural network with different parameters and the passage of points from 1 to the last. The choice of MLP architecture for a specific practical problem should be determined by the desired quality of classification and the analogy with the results obtained by other researchers for similar problems. At the same time, it is recommended to choose a larger number of hidden layers and neurons in them, in comparison with analogues. This will allow to obtain a higher quality of image classification with the required amount of calculations as a result of pruning according to this technique:

1. Choice of the initial step of learning. To choose the initial step of learning, it is possible to studied in this article contained 234752 weight coefficients. The best results were obtained with a learning step of 0.005. Based on this, it seems appropriate to choose the learning step inversely proportional to the increase in the number of weights in the original neural network that needs to be learned.

2. Choice of initial values for weight coefficients. If a sigmoid is used as an activation function, it is advisable to use Xavier's initialization [31] for initialization. In the initial state of the neural network, all connections between layers are present. The weight coefficients are given as random numbers with a normal distribution, zero mean, and variance inversely proportional to the number of connections included in the corresponding neuron.

If another function is used as an activation function, then recommendations [31] should be used.

3. Choice of the pruning interval and the number of links to be deleted at a time. Based on Fig. 6, *b*, it is advisable to choose the pruning interval – one epoch, the number of links removed at a time – one percent.

4. Perform learning with the selected learning step, with smaller and larger ones. This will determine the success of choosing the initial value for the learning step. If the learning curve has a form characteristic of the degradation of the learning process (this was at a learning step of 0.02 and 0.04), then it is necessary to learn with a smaller step.

If there is an improvement in the classification with an increase in the learning step, then it must be increased until this leads to a result.

When learning, it is necessary to use the preprocessing of input signals described in this article.

5. Perform learning with a smaller percentage of deleted links. As can be seen from Fig. 4, 5, an excessively large amount of links removed during pruning leads to a decrease in potentially achievable characteristics. On the other hand, too few links removed at one time will significantly lengthen the total time to find the required configuration. Therefore, it is necessary to repeatedly learn the neural network with an ever smaller percentage of removed connections until this leads to an improvement in recognition on the learning set.

6. Analysis of the total percentage of remote connections until a sign appears that all redundancy has been removed from the neural network. If when learning a neural network a low percentage of limit pruning is obtained, for example, 10–15 %, then it is recommended to add neurons, increase the number of hidden layers, and repeat all the points, starting from 1. The final percentage of limit pruning should be about 80–95 %.

6. Discussion of the results of studying the influence of the choice of parameters of the pruning procedure on the learning quality of a multilayer perceptron

Neural network learning is an iterative solution of a system of non-linear algebraic equations of high dimension. This is a difficult mathematical problem. Moreover, the result essentially depends on the configuration of the neural network and the class of signals that are input. With an unchanged configuration of the neural network, the connections between neurons remain constant, only the values of the weight coefficients change. When there is redundancy in the neural network, or it is added, there is an additional degree of freedom in that some connections can be removed. This makes it possible to provide a better correspondence between the neural network configuration and the class of input signals, which gives a higher quality of classification [29].

The results of the experimental study in Fig. 6 show that the dependence of the resulting classification quality from the parameters of the pruning procedure has a convex character. This made it possible to formulate a methodology for searching for parameters that provide the highest quality of classification. In addition, the resulting typical learning curves shown in Fig. 2–5 allow to navigate with respect to the learning mode of the neural network at the moment and, accordingly, change the parameters. An analysis of typical learning curves makes it possible to clearly identify the complete removal of redundancy in the neural network, after which pruning only leads to a deterioration in the quality of the classification.

A feature of the results obtained is that they were obtained on a single MLP. This was done in order to determine in its pure form the effect of the parameters of the pruning procedure on the MLP. Therefore, the numerical values of the indicators refer specifically to the single used MLP. The general patterns identified for these uses will hold true for more complex use cases. However, the contribution of MLP to the resulting characteristics requires further research. Such applications include, for example, MLP at the output of a convolutional neural network or numerous MLPs as part of modern architectures [2–9].

7. Conclusions

1. Learning with pruning of a multilayer perceptron using the example of image classification from the MNIST set with a wide range of values for the parameters of the pruning procedure shows the possibility of increasing the classification quality by about 1 %. At the same time, the initial number of connections 234752 during learning is reduced by 80 %. The best results are observed when removing 1 % of the connections with the smallest units of weights after each learning epoch. In addition, the study shows a convex nature of the dependence of the classification quality on the values of the parameters of the learning procedure with pruning.

2. A technique for finding the best parameters of the learning procedure with link pruning in a multilayer perceptron is formulated, which can be used for any set of input images and any MLP architecture. The use of the technique allows not to do a complete enumeration of all possible variants of the values of the learning procedure with pruning, but to use the minimum set of steps to achieve the best classification quality. The steps of the technique are aimed at achieving a balance between the learning rate, the size of the pruning interval, and the number of links removed at a time.

References

1. Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. Neurocomputing, 234, 11–26. doi: http://doi.org/10.1016/j.neucom.2016.12.038
2. Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T. et. al. (2021). MLP-Mixer: An all-MLP Architecture for Vision. ArXiv. Available at: https://arxiv.org/abs/2105.01601
3. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T. et. al. (2021). An image is worth 16×16 words: transformers for image recognition at scale. ArXiv. Available at: https://arxiv.org/abs/2010.11929
4. Hassani, A., Walton, S., Shah, N., Abuduweili, A., Li, J., Shi, H. (2021). Escaping the Big Data Paradigm with Compact Transformers. ArXiv. Available at: https://arxiv.org/abs/2104.05704
5. Patches Are All You Need? (2021). Under review as a conference paper at ICLR 2022. Available at: https://openreview.net/pdf?id=TVHS5Y4dNvM
6. Guo, M.-H., Liu, Z.-N., Mu, T.-J., Hu, S.-M. (2021). Beyond Self-attention: External Attention using Two Linear Layers for Visual Tasks. ArXiv. Available at: https://arxiv.org/abs/2105.02358
7. Lee-Thorp, J., Ainslie, J., Eckstein, I., Ontañón, S. (2021). FNet: Mixing Tokens with Fourier Transforms. ArXiv. Available at: https://arxiv.org/abs/2105.03824
8. Liu, H., Dai, Z., So, D. R., Le, Q. V. (2021). Pay Attention to MLPs. ArXiv. Available at: https://arxiv.org/abs/2105.08050
9. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B. (2021). Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. ArXiv. Available at: https://arxiv.org/abs/2103.14030
10. Denil, M., Shakibi, B., Dinh, L., Ranzato, M. A., Freitas, N. (2014). Predicting Parameters in Deep Learning. ArXiv. Available at: https://arxiv.org/abs/1306.0543
11. Blalock, D., Gonzalez Ortiz, J. J., Frankle, J., Guttag, J. (2020). What is the state of neural network pruning? ArXiv. Available at: https://arxiv.org/abs/2003.03033

12. Han, S., Pool, J., Tran, J., Dally, W. J. (2015). Learning bothWeights and Connections for Efficient Neural Networks. ArXiv. Available at: https://arxiv.org/pdf/1506.02626v3.pdf
13. LeCun, Y., Denker, J. S., Solla, S. A. (1990). Optimal Brain Damage. NIPS. Available at: http://yann.lecun.com/exdb/publis/pdf/lecun-90b.pdf
14. Hinton, G., Vinyals, O., Dean, J. (2015). Distilling the knowledge in a neural network. NIPS Deep Learning and Representation Learning Workshop. ArXiv. Available at: https://arxiv.org/abs/1503.02531
15. Li, C., Peng, J., Yuan, L., Wang, G., Liang, X., Lin, L., Chang, X. (2020). Block-Wisely Supervised Neural Architecture Search With Knowledge Distillation. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). doi: http://doi.org/10.1109/cvpr42600.2020.00206
16. Aflalo, Y., Noy, A., Lin, M., Friedman, I., Zelnik, L. (2020). Knapsack Pruning with Inner Distillation. ArXiv. Available at: https://arxiv.org/abs/2002.08258
17. Wang, Z., Li, F., Shi, G., Xie, X., Wang, F. (2020). Network pruning using sparse learning and genetic algorithm. Neurocomputing, 404, 247–256. doi: http://doi.org/10.1016/j.neucom.2020.03.082
18. Denton, E. L., Zaremba, W., Bruna, J., LeCun, Y., Fergus, R. (2014). Exploiting linear structure within convolutional networks for efficient evaluation. Advances in Neural Information Processing Systems, 1269–1277.
19. Li, Y., Gu, S., Mayer, C., Van Gool, L., Timofte, R. (2020). Group Sparsity: The Hinge Between Filter Pruning and Decomposition for Network Compression. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). doi: http://doi.org/10.1109/cvpr42600.2020.00804
20. Han, S., Mao, H., Dally, W. J. (2015). Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding. ArXiv. Available at: https://arxiv.org/abs/1510.00149
21. Qiu, J., Wang, J., Yao, S., Guo, K., Li, B., Zhou, E. et. al. (2016). Going Deeper with Embedded FPGA Platform for Convolutional Neural Network. Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. Monterey. doi: http://doi.org/10.1145/2847263.2847265
22. Paupamah, K., James, S., Klein, R. (2020). Quantisation and Pruning for Neural Network Compression and Regularisation. 2020 International SAUPEC/RobMech/PRASA Conference. doi: http://doi.org/10.1109/saupec/robmech/prasa48453.2020.9041096
23. Louizos, C., Welling, M., Kingma, D. P. (2018). Learning sparse neural networks through l0 regularization. ICLR 2018. ArXiv. Available at: https://arxiv.org/abs/1712.01312
24. Li, J., Qi, Q., Wang, J., Ge, C., Li, Y., Yue, Z., Sun, H. (2019). OICSR: Out-In-Channel Sparsity Regularization for Compact Deep Neural Networks. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). doi: http://doi.org/10.1109/cvpr.2019.00721
25. Gomez, A. N., Zhang, I., Kamalakara, S. R., Madaan, D., Swersky, K., Gal, Y. et. al. (2019). Learning Sparse Networks Using Targeted Dropout. ArXiv. Available at: https://arxiv.org/abs/1905.13678
26. Tabik, S., Peralta, D., Herrera-Poyatos, A., Herrera, F. (2017). A snapshot of image pre-processing for convolutional neural networks: case study of MNIST. International Journal of Computational Intelligence Systems, 10 (1), 555–568. doi: http://doi.org/10.2991/ijcis.2017.10.1.38
27. Cireşan, D. C., Meier, U., Gambardella, L. M., Schmidhuber, J. (2010). Deep, Big, Simple Neural Nets for Handwritten Digit Recognition. Neural Computation, 22 (12), 3207–3220. doi: http://doi.org/10.1162/neco_a_00052
28. Simard, P. Y., Steinkraus, D., Platt, J. C. (2003). Best practices for convolutional neural networks applied to visual document analysis. Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings. San Mateo: IEEE Computer Society Press, 958–962. doi: http://doi.org/10.1109/icdar.2003.1227801
29. Galchonkov, O., Nevrev, A., Glava, M., Babych, M. (2020). Exploring the efficiency of the combined application of connection pruning and source data preprocessing when training a multilayer perceptron. Eastern-European Journal of Enterprise Technologies, 2 (9 (104)), 6–13. doi: http://doi.org/10.15587/1729-4061.2020.200819LeCun, Y., Cortes, C., Burges, C. J. C. The mnist database of handwritten digits. Available at: http://yann.lecun.com/exdb/mnist/
30. Brownlee, J. (2021). Weight Initialization for Deep Learning Neural Networks. Available at: https://machinelearningmastery.com/weight-initialization-for-deep-learning-neural-networks/
31. Colab. Available at: https://colab.research.google.com/notebooks/welcome.ipynb