UDC 681.3.07: 004.8
DOI: 10.15587/1729-4061.2020.200819

EXPLORING THE EFFICIENCY OF THE COMBINED APPLICATION OF CONNECTION PRUNING AND SOURCE DATA PRE-PROCESSING WHEN TRAINING A MULTILAYER PERCEPTRON

O. Galchonkov
PhD, Associate Professor*
E-mail: o.n.galchenkov@gmail.com
A. Nevrev
PhD, Associate Professor*
E-mail: a.i.nevrev@gmail.com
M. Glava
PhD, Associate Professor*
E-mail: glavamg@gmail.com
M. Babych
PhD, Associate Professor*
E-mail: babich.tiger@gmail.com
*Department of Information Systems
Institute of Computer Systems
Odessa National Polytechnic University
Shevchenka ave., 1, Odessa, Ukraine, 65044

Традиційною схемою роботи з нейронними мережами до недавнього часу було завдання архітектури нейронної мережі та її подальше навчання. Однак останні дослідження в цій області показали, що задані і налаштовані таким чином нейронні мережі мають значну надмірність. Тому додатковою операцією стала ліквідація цієї надмірності за рахунок проріджування зв'язків в архітектурі нейронної мережі. Серед безлічі підходів до ліквідації надмірності найбільш перспективним представляється спільне використання декількох методів, коли їх сумарний ефект перевищує суму ефектів одиночного використання кожного з них. Проведено експериментальне дослідження ефективності спільного використання ітераційного проріджування і предобробки (предіскажeній) вхідних даних в задачі розпізнавання рукописних цифр за допомогою багатошарового персептрона. Показано, що використання предобробки вхідних даних регулярізує процедуру навчання нейронної мережі, запобігаючи її перенавчання. Спільне використання ітераційного проріджування і предобробки вхідних даних дозволило отримати меншу помилку розпізнавання рукописних цифр – 1,22 %, в порівнянні з використанням тільки проріджування (помилка зменшилася з 1,89 % до 1,81 %) і з використанням тільки предіскажeній (помилка зменшилася з 1,89 % до 1,52 %). Крім цього регуляризація за рахунок предіскажeній дозволяє отримати монотонно збільшуючуюся кількість відключених зв'язків при збереженні помилки на рівні 1,45 %. Отримані криві навчання для однієї і тієї ж задачі, які відповідають початку навчання з різних початкових умов, мають різні значення як в процесі навчання, так і в кінці навчання. Це свідчить про багатоекстремальність функції якості – точності розпізнавання. Практичне використання цього полягає в пропозиції проводити багаторазове навчання нейронної мережі з вибором найкращого результату

***Ключові слова:** багатошаровий персептрон, нейронна мережа, проріджування, регуляризація, крива навчання, вагові коефіцієнти*

*Received date 23.02.2020
Accepted date 10.04.2020
Published date 30.04.2020*

1. Introduction

Deep neural networks are a powerful tool for addressing a wide range of tasks in the fields of image processing, unmanned object management, disease diagnostics, recognition, voice signal generation, etc. [1]. The efficiency of the practical use of neural networks improves as their dimensionality increases. In this case, a practical constraint is the limited computational power of the processors used. This is especially true when a mobile phone or microcontroller serves as a computational tool. This circumstance has initiated a large body of research that identified the redundancy of fully connected neural networks [2] and defined basic approaches to reducing it. For example, it is shown in [3] that for the popular neural networks AlexNet (more than 61 million weights) and VGG-16 (more than 132 million weights) employing the ImageNet dataset, a decrease in the number of weight coefficients by 9 and 13 times, respectively, could be achieved without compromising their performance quality. Despite the progress made, it is a relevant task to undertake further research aimed at reducing the redundancy of neural networks while maintaining, or even improving, their operational quality.

2. Literature review and problem statement

The main characteristics of neural networks are the quality of operation and the amount of computation required to implement them. Typically, research is aimed at improving only one of these characteristics.

One of the first approaches to reducing the redundancy of neural networks was proposed in [4, 5]. It implies pre-training a fully-connected neural network, then reducing the excess connections and retraining the pruned neural network. It was shown that it is possible to reduce the volume of computations for the implementation of a trained neural network by 60 % even without worsening the quality of its operation. Subsequent studies focused on further reducing both the amount of computation by a trained neural network and the amount of computation to train it at the same or slightly worse quality of operation. Paper [3] proposed an iterative procedure for a step-by-step weight reduction and a neural network post-training, which made it possible to reduce by many times the number of weight coefficients in the trained neural network.

Study [6] uses a low-ranking approach method; the variation dropout was employed in [7]. The proposed algorithms made it possible to speed up the training and reduce the volume of computations during training. Work [8] proposed a method of deep compression consisting of three stages: the usual reduction of connections, similarly to [3], the quantization of weights, and Huffman coding. The further adaptation of neural networks to the limited computing capabilities of a controlling element is reported in [9]. The cited paper proposes to use the quantization of input signals with a dynamically changing accuracy. However, the algorithms proposed in [8, 9] could be implemented effectively only by using programmable logic matrices and are not suitable for conventional processors.

The approaches in [3–9] are characterized by that the learning process begins in a fully-connected topology of the neural network, followed by a decrease in the number of connections in the course of training. It is possible to achieve a significant reduction in the required volume of RAM, in energy consumption, and the required number of arithmetic operations, by pruning the structure of a neural network at the outset, to then train such a pruned network. Paper [10] compares the characteristics of neural network training that begin learning with a fully connected topology and the RadiX-Nets-type networks that initially have a pruned structure. It was shown that the same end characteristics are obtained under such an approach but the individual implementations of the RadiX-Nets training could demonstrate a computational instability. Addressing this instability is the direction for further research. A similar approach was applied in [11]; it is proposed to prune a neural network before training by using a connection sensitivity criterion. It is possible to further train the resulting pruned network using any modern algorithm. Such an approach makes it possible to achieve the weight coefficient saving of up to 99 % in known neural networks but this would be accompanied by a deterioration in the quality of their performance.

The IPLT (incremental pruning based on less training) approach, proposed in [12], can serve an interim solution. While the traditional approach, used, for example, in [3], implies a long-time neural network training at every step and a slight reduction of its connections, the approach in [12] employs small learning intervals followed by a radical reduction of its connections. Achieving the desired reduction in the configuration of the neural network is followed by training it to the established mode. Such an approach ensures the same quality characteristics as the conventional one but the rate of learning increases by an order of magnitude. It should be noted that a neural network can be trained using a supercomputer while its implementation in the trained form – using a low-power microprocessor. Therefore, the rate of learning is typically not a critical parameter for neural networks implemented on low-power computers.

The approaches reported in [3–12] are aimed primarily at reducing the redundancy of neural networks while maintaining or even compromising the operational quality of neural networks. This makes it easier to implement such solutions on less-performing computing devices but does not improve the quality of their operation.

In terms of improving the quality of neural network performance, the most promising are those approaches that are orthogonal to each other. The combined application of such approaches provides a new quality. A prime example is the computational eco-system [13], which achieved the best result when recognizing handwritten digits from the MNIST set [14] – the recognition error was 0.17 %. This is 0.03 % better than the result shown by a person (0.2 %). The topology of this network contains a set of diverse classifiers built on the basis of the convolutional neural networks, which, in turn, employ different approaches for learning. All these networks run in parallel and their outputs are the inputs of the unifying Meta-Nets network. A key factor in this result is a variety of approaches for training the primary convolutional neural networks.

It should be noted that the improvement in the operational quality in [13] was achieved by repeatedly increasing the volume of computations by the trained neural network. Therefore, in order to reduce the redundancy of neural networks while improving the quality of operation, it seems appropriate to study approaches that differ from each other in the principle of action and the possibility of their combined application. Among these directions is the use of an input signal distortion and its joint work with the usual pruning [3] regarding the training of a multilayer perceptron. On the one hand, the multilayer perceptron is part of the convolutional networks as an element of the finishing processing. On the other hand, it has a rather simple and homogeneous topology, which makes its use easier in the analysis and signal processing peripheral devices, implemented on the microcontrollers – smart sensors.

The studies reported in [15–17] show that the pre-distortion of input images in the form of turns, horizontal and vertical shifts, compression, as well as stretching horizontally and vertically, make it possible to significantly improve the characteristics of object recognition on images. The impact of each separate pre-distortion on the resulting characteristics of such known convolutional networks as LeNet, Network3, and DropConnect was investigated in [15]. It was shown that the use of each type of predistortion leads to an improvement in the characteristics of a neural network but the greatest effect is obtained at a combined application of all kinds of pre-distortions. Paper [16] demonstrated that the pre-distortion of input images from the MNIST set when using a fully-connected multilayer perceptron with a number of hidden layers from 2 to 9 and a total quantity of the weight coefficients from 1.3 to 12 million reduces the error of handwritten digit recognition to 0.35 %. A comparison of the effectiveness of the use of the convolutional neural networks and a multilayer perceptron for analyzing text documents involving pre-distortions using the MNIST dataset as an example was carried out in [17]. It was shown that even simple algorithms to train a neural network with a simple topology, a multilayer perceptron, can ensure a small recognition error when trained. In this case, it may take a large number of epochs to learn but, given today's computational capabilities, this is not a problem. It was noted that an essential factor that ensures a small magnitude of the recognition error is the quality, quantity, and variety of input data, which, in turn, can be acquired by pre-distorting the source data. All the above studies have shown the effectiveness of using the pre-processing of input data to improve the quality of neural networks operation. However, they did not aim at reducing the redundancy of weight coefficients.

Given the effectiveness of applying the approaches to prune neural networks and to pre-distort the input data, it is interesting to study their combined application in the training of a multilayer perceptron using one of the most popular data sets, MNIST.

3. The aim and objectives of the study

The aim of this study is to examine the effectiveness of the combined application of data pre-distortion and the iterative algorithm to reduce connections and to post-train a multilayer perceptron using an example of handwritten digit recognition from the MNIST set.

To accomplish the aim, the following tasks have been set:

– to identify the differences in training a multilayer perceptron in the presence and absence of connection pruning provided that the pre-distortion is not used in this case;

– to compare the learning curves and the resulting characteristics of a multilayer perceptron in the presence and absence of input data pre-processing provided that the pruning is not used in this case;

– to identify the effectiveness of the combined application of data pruning and pre-distortion.

4. Parameters for the applied neural network and input data

4. 1. Description of a neural network and its training algorithm

The original neural network used was a perceptron with two hidden layers. The first hidden layer contained 256 neurons, the second hidden layer – 128 neurons. The function of neuron activation is sigmoid. The input layer had 784 neurons (by the number of pixels in the images of handwritten digits from the MNIST set). The output layer had 10 neurons, each of which was assigned the corresponding number from a series of 0, ..., 9. The simplest algorithm of stochastic gradient descent (SGD) with a constant step was used as a training algorithm [18]. The initial values for the neural network's weight coefficients were set by a random number generator at normal distribution, a zero average, and an rms deviation whose magnitude is inversely proportional to the root square from the number of connections included in a neuron.

The following designations were introduced:

– $n_{in}=784$ – the number of nodes in the input layer of the neural network (the number of inputs to a neural network);

– $n_{h1}=256$ – the number of neurons in the first hidden layer;

– $n_{h2}=128$ – the number of neurons in the second hidden layer;

– $n_o=10$ – the number of neurons in the output layer (the number of outputs from a neural network);

– X_{in} – the vector of input signals to the neural network, dimensionality n_{in};

– X_{in-h1} – the vector of input signals to the first hidden layer, dimensionality n_{h1};

– X_{o-h1} – the vector of output signals from the second hidden layer, dimensionality n_{h1};

– X_{in-h2} – the vector of input signals to the second hidden layer, dimensionality n_{h2};

– X_{o-h2} – the vector of output signals from the second hidden layer, dimensionality n_{h2};

– X_{in-o} – the vector of input signals to the output layer, dimensionality n_o;

– X_{out} – the vector of output signals from the output layer (an output signal from a neural network), dimensionality n_o;

– X_{tar} – the vector of known correct output signals from the neural network, dimensionality n_o;

– W_{in-h1} – the matrix of weights between the input layer and the first hidden layer, dimensionality $n_{h1}\times n_{in}$;

– W_{h1-h2} – the matrix of weights between the first and second hidden layers, dimensionality $n_{h2}\times n_{h1}$;

– W_{h2-o} – the matrix of weights between the second hidden layer and the output layer, dimensionality $n_o\times n_{h2}$;

– E_{out} – the vector of an error at the output from the neural network, dimensionality n_o;

– E_{h2} – the vector of an error at the output from the second hidden layer, dimensionality n_{h2};

– E_{h1} – the vector of an error at the output from the first hidden layer, dimensionality n_{h1}.

The input and output signals vectors for the hidden layers and an output layer:

$$X_{in-h1}=W_{in-h1}\times X_{in}, \quad (1)$$

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

where $f_{act}(x)=1\big/\left(1+e^{-x}\right)$ is the activation function,

$$X_{in-h2}=W_{h1-h2}\times X_{o-h1}, \quad (3)$$

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

$$X_{in-o}=W_{h2-o}\times X_{o-h2}, \quad (5)$$

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

The vectors of errors at the outputs from the hidden layers and an output layer:

$$E_{out}=X_{tar}-X_{out}, \quad (7)$$

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

$$E_{h1}=W_{h1-h2}^{T}\times E_{h2}. \quad (9)$$

The renewal of the neural network's weight coefficients is performed according to the following equations:

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

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

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

where the «·» operation denotes an element-wise multiplying; $\mu=0.01$ is the step of training, a scalar value.

The proposed modification of the algorithm implies that the weight coefficients update is supplemented by the following equations:

$$W_{h2-o} = W_{h2-o} \cdot H_{h2-o}, \qquad (13)$$

$$W_{h1-h2} = W_{h1-h2} \cdot H_{h1-h2}, \qquad (14)$$

$$W_{in-h1} = W_{in-h1} \cdot H_{in-h1}, \qquad (15)$$

where the H_{h2-o}, H_{h1-h2}, H_{in-h1} matrices have, respectively, the dimensionalities of $n_o \times n_{h2}$, $n_{h2} \times n_{h1}$, $n_{h1} \times n_{in}$.

At the beginning of training, all elements in the H_{h2-o}, H_{h1-h2}, H_{in-h1} matrices accept single values. At the end of each subsequent L epoch of training, the weight coefficients W_{in-h1}, W_{h1-h2}, W_{h2-o} matrices are analyzed. If any ij elements of these matrices accept a module below the corresponding threshold P_{in-h1}, P_{h1-h2}, P_{h2-o}, then the H_{h2-o}, H_{h1-h2}, H_{in-h1} matrices elements are assigned a value of 0.

4. 2. Description of the MNIST dataset

MNIST dataset is one of the main tests of different neural network structures and their training algorithms [14].

The MNIST dataset consists of two parts. The first subset consists of 60,000 images of handwritten digits from 0 to 9, designed to train a neural network. The second subset contains 10,000 similar images for testing. The task is to use data for learning only to train a neural network to recognize digits from the subset for testing. Each image is 28×28 pixel-sized and is accompanied by information about the digit in a given image. The color of the images is black and white. The color of each pixel in the image is encoded by an integer eight-bit binary number ranging from 0 (black) to 255 (white). The typical images of the digits are shown in Fig. 1.

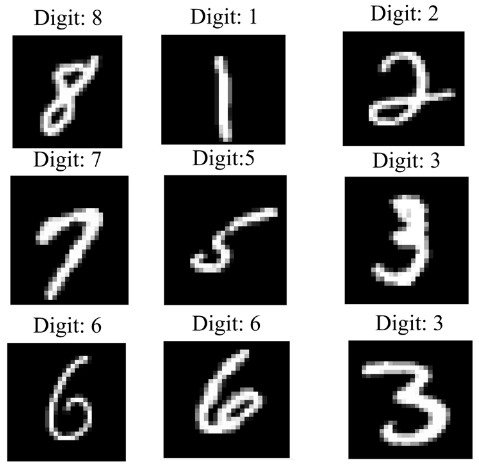

Fig. 1. Typical images of digits from the MNIST set

When operating the MNIST dataset, a neural network must have $n_{in} = 784$ inputs (based on the number of pixels in the image) and $n_o = 10$ outputs, each of which is assigned its own digit. X_{tar}, the vector of known correct output signals for the set image must accept 1 at the input at a place corresponding to the digit shown in this image, and 0 at the remaining ones. For example, if an image of a four is sent to the input,

$$X_{tar} = (0\,0\,0\,0\,1\,0\,0\,0\,0\,0)^T. \qquad (16)$$

The response of the neural network is determined by the position of the maximal element in the X_{out} vector.

$$\text{figure in the image} = \arg\max_i \{x_{out}[i]\}. \qquad (17)$$

To train a neural network, the output signal vector X_{out} is used in full according to equations (1) to (16).

4. 3. Data preprocessing

The following operations were used to pre-process the data:

– the rotation of an image relative to the center, the range of rotation angles is from −15 degrees to +15 degrees, the specific value for each operation is set by its random number generator with an even distribution;

– a horizontal shift in the range of –0.05 to 0.05 from the size of an image for width, the specific value of the shift magnitude is set by its random number generator with an even distribution;

– a vertical shift in the range of –0.05 to 0.05 from the size of an image for height, the specific value of the shift magnitude is set by its random number generator with an even distribution;

– a change in the size of an image by a factor ranging from 0.95 to 1.05 from the original image size, the specific value of the shift magnitude is set by its random number generator with an even distribution;

– filling the free image pixels with the same values as the nearest filled pixels (for example, when shifting the image to 1 pixel to the right, a series of unfilled pixels are formed on the left, which are filled with pixel values from the penultimate left row);

– an increase in the contrast after the above operations, if the pixel value exceeds 100, it is assigned a value of 255 (white), if the pixel value is less than or equal to 100, it is assigned a value of 0 (black).

Before submitting the next image, intended for training, from the MNIST set to the neural network input, all operations from the described set are consistently performed over it. This ensures the uniqueness of each of the images coming to the neural network input. Images from the test set are not processed.

5. Results of an experimental study of the effectiveness of using the pruning and pre-distortion of data in the training of a multilayer perceptron

5. 1. Studying the differences in multilayer perceptron training in the presence and absence of connection pruning

Fig. 2 shows the learning curves of a two-layer perceptron. Each curve corresponds to training a neural network under the new random initial conditions for weight coefficients. Curves 1–3: no weight coefficient zeroing was used. Curves 4–6: the weight coefficients were zeroed following each training epoch. Fig. 3 shows the charts of change in the proportion of zero coefficients corresponding to curves 4–6 in Fig. 2.

One epoch corresponded to the training on 60,000 images, intended for training, from the MNIST set. Each point on the learning curves was derived in the following way. After the training based on every 1,000 images, we estimated the accuracy of digit recognition on a set of 10,000 images for testing from the MNIST set. In this case, of course, the permanent weights were used, whose values were established at a given moment. Next, we determined, from a set of 180 recognition accuracy values, sequentially derived in this way, the maximum one, which was used for a value on the learning curve. Thus, each point on the learning curve corresponds to the maximum value of accuracy achieved during the three training epochs. The weight set that yielded this maximum value is remembered and can be used in the further practical use of a trained neural network.

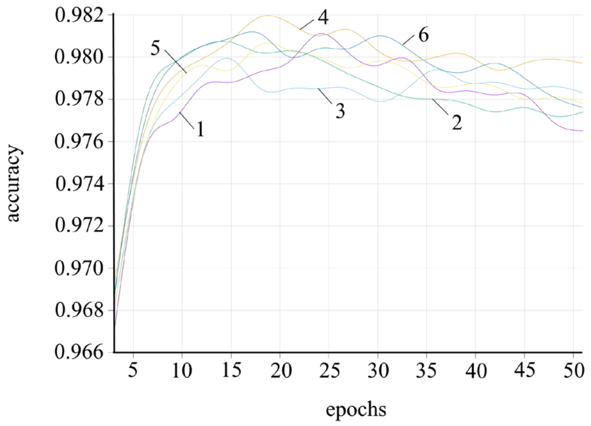

Fig. 2. The learning curves of a two-layer perceptron under different initial conditions: 1, 2, 3 – without weight coefficients zeroing; 4, 5, 6 – weight coefficients were zeroed after each training epoch

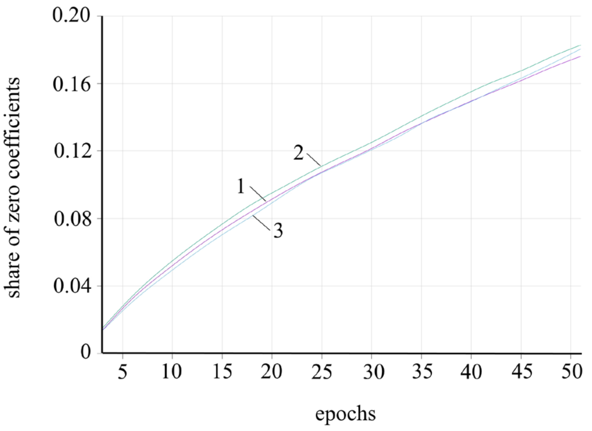

Fig. 3. Charts of change in the proportion of zero weight coefficients depending on the number of a training epoch. Curves 1–3 correspond to curves 4–6 in Fig. 2

The learning curves 4–6 in Fig. 2 were received using the following thresholds:

$$P_{in-h1} = w_{in-h1}^{\max} / 1000, \qquad (18)$$

$$P_{h1-h2} = w_{h1-h2}^{\max} / 4000, \qquad (19)$$

$$P_{h2-o} = w_{h2-o}^{\max} / 8000, \qquad (20)$$

where $w_{in-h1}^{\max}$ is the maximum module of the current elements of the W_{in-h1} matrix; $w_{h1-h2}^{\max}$ is the maximum module of the current elements of the W_{h1-h2} matrix; $w_{h2-o}^{\max}$ is the maximum module of the current elements of the W_{h2-o} matrix.

Fig. 2 shows that the use of weight zeroing operation does not impair the results of digit recognition in the images from the test set. Moreover, the best result without zeroing is 0.9811, with it – 0.9819. In this case, the weight set, which ensured the recognition accuracy of 0.9819, contained more than 8.4 % of zero weight coefficients.

Fig. 4 shows the learning curves for a similar neural network but the first hidden layer contained 8 % fewer neurons, that is, 235. No zeroing was used.

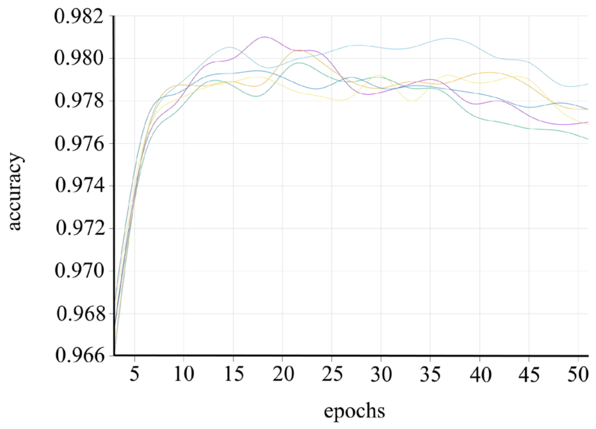

Fig. 4. The learning curves of a two-layer perceptron under various random initial conditions without zeroing the weights, the number of neurons in the first hidden layer is 235, in the second – 128

A comparison of the learning curves in Fig. 2 and 4 shows a slight deterioration in the quality of digit recognition in the images when reducing the number of neurons in the first hidden layer by 8 %.

The range of curves 1–3 maxima in Fig. 2 – 0.9811–0.9799.

The range of curves 4–6 maxima in Fig. 2 – 0.9819–0.9806.

The range of curves 1, ..., 6 maxima in Fig. 4 – 0.981–0.9791.

Thus, the training algorithm with zeroing ensures an improvement in the quality of recognition while reducing the number of weight coefficients. At the same time, a simple decrease in the number of weight coefficients leads to a deterioration in the quality of recognition.

The charts in Fig. 3 show a monotonous increase in the proportion of zero weights depending on the number of training epochs, which, for example, for curve 5, accepts a value exceeding 18 % after 51 training epochs.

However, Fig. 2 demonstrates that the neural network is retrained after 18–30 epochs, that is, the accuracy of digit recognition in the images decreases. Therefore, in actual practice, the weight coefficient set to be used should be chosen based on the joint requirements for the accuracy of recognition and the proportion of zero weights. All weight sets obtained in the course of training can be stored in memory with any of them to be employed in the implementation of a neural network.

5. 2. Studying the differences in training a multilayer perceptron in the presence and absence of the source data pre-processing

Fig. 5 shows the learning curves of a multilayer perceptron (256 neurons in the first hidden layer and 128 neurons in the second hidden layer).

The training was each time performed under the new random initial conditions for weight coefficients. Curves 1–3: without using the predistortion of starting images; curves 4–6: applying it.

A comparison of these learning curves unambiguously testifies to the positive effect of using the predistortion during training.

The range of curves 1–3 maxima in Fig. 4 – 0.9811–0.9799. For all curves 4–6 in Fig. 4, a maximum is 0.9841.

In addition, while curves 1–3 clearly show the effect of retraining, expressed by the descent in the curves after 18–30 epochs, curves 4–6 demonstrate no effect of retraining. This suggests that the predistortions used act as a regularizer.

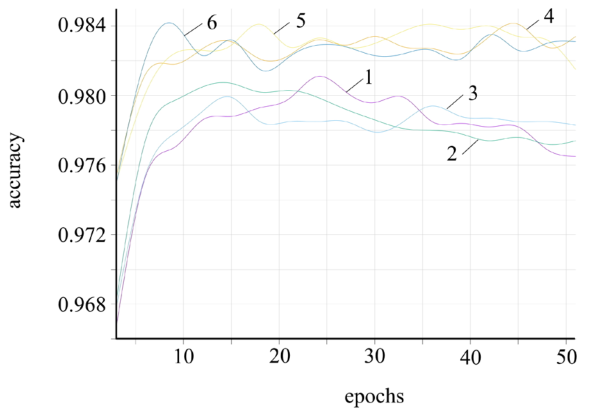

Fig. 5. The learning curves of a multilayer perceptron under different random initial conditions: 1, 2, 3 — without predistortion; 4, 5, 6 — with predistortion

Fig. 6 shows the separate detailed sections of the learning curves based on the training data and the test data results. The values ik are arranged along the horizontal axis, each unit of the ik parameter corresponds to the training on the next 1,000 images at the input to the neural network; along the vertical axis – the accuracy of recognition.

Fig. 6, *a, c, e, g* shows that the retraining of the neural network manifests itself in the degeneration of the learning curve based on training data into a straight line. At the same time, the use of pre-processing (Fig. 6, *b, d, f, h*) results in that the learning curve based on the training data acquires a constantly quasi-noisy shape. That is, there is a constant process of learning based on the new data at the input to the neural network.

5. 3. Exploring the effectiveness of the combined application of pruning and pre-distortion of data

Fig. 7 shows the learning curves of the multilayer perceptron (256 neurons in the first hidden layer and 128 neurons in the second hidden layer). The training was each time performed under the new random initial conditions for weight coefficients. Curves 1–3: using only the predistortion of starting images; curves 4–6: using the predistortion and pruning.

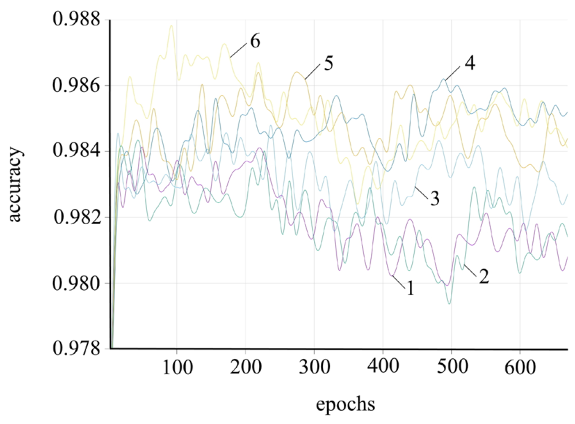

Fig. 7. The learning curves of a multilayer perceptron under different random initial conditions. 1–3: using only the predistortions; 4–6: using the predistortion and pruning after each training epoch

A comparison of the learning curves in Fig. 7 shows that the combined application of the iterative pruning and pre-processing of source data has resulted in the maximum recognition accuracy of 0.9878, which corresponds to a recognition error of 1.22 %. At the same time, the use of pruning only (Fig. 2) reduced the error from 1.89 % to 1.81 %, while using the predistortion only (Fig. 5, 7) reduced the error from 1.89 % to 1.52 %.

Thus, the combined application of the iterative pruning and pre-processing of source data results in a greater effect in terms of recognition accuracy than the sum of the effectiveness of the separate use of pruning and pre-distortion.

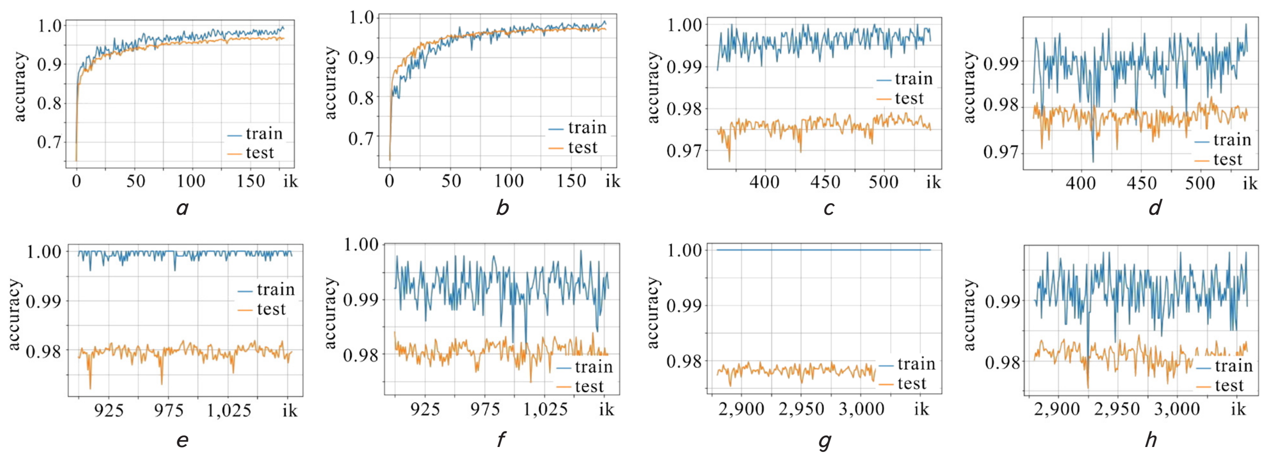

Fig. 6. The fragments of learning curves based on training data and for test data: *a, c, e, g* — without the distortion of starting images, *b, d, f, h* — using the predistortion; *a* — ik=1...175, *b* — ik=1...175, *c* — ik=350...550, *d* — ik=350...550, *e* — ik=900...1,075, *f* — ik=900...1,075, *g* — ik=2,875...3,050, *h* — ik=2,875...3,050

In addition, it is necessary to note the regularizing effect from using the predistortion, which makes it possible to obtain a monotonous reduction in the number of connections in a neural network while maintaining a recognition error at the level of 1.45 % or less throughout 660 training epochs. Fig. 8 shows the charts of change in the proportion of zero weight coefficients depending on the number of a training epoch, corresponding to curves 4–6 in Fig. 7.

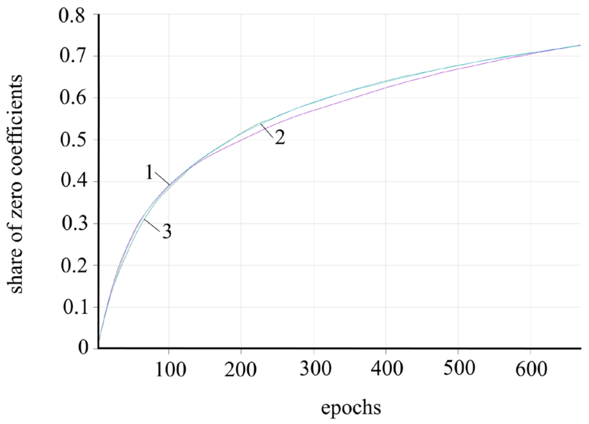

Fig. 8. Charts of change in the proportion of zero weight coefficients depending on the number of a training epoch. Curves 1–3 correspond to curves 4–6 in Fig. 7

Fig. 2, 3 show that the pure pruning (zeroing the weights) led, after training during 51 epochs, to an increase in the error of recognition to 2 % and higher at the number of zero coefficients of about 18 %. At the same time, Fig. 7, 8 show that over 660 epochs the number of zero coefficients reached 72 % of the initial number of weight coefficients in a fully-connected neural network while the recognition error remained at 1.45 % or less.

6. Discussion of results of studying the effectiveness of using the pruning and pre-distortion of data in the training of a multilayer perceptron

Our experimental study of the combined application of pruning and predistortions has shown that their joint application is efficient and produces an additional effect on the accuracy of handwritten digits recognition, as well as maintains the recognition error at a low level when pruning the connections in a multilayer perceptron. The study was conducted based on the MNIST dataset. One can expect that in tasks close to the set of handwritten digits MNIST, the would-be effect would be of the same order. At the same time, the regularizing properties of the input data predistortions could be used in a wide range of neural network learning tasks along with regularization methods such as L2, Dropout, and others [1].

The learning curves in Fig. 2, 4, 5, 7 show that the process of training a multilayer perceptron is significantly dependent on the initial conditions for weight coefficients, which is the result of the complex nature of the surface of the quality function – the accuracy of recognition. This fact can be practically implemented in two ways. First, by conducting multiple neural network training, it becomes possible, starting from different random initial conditions, to choose a variant of weight coefficients that would yield a minimal error. Second, the neural network variants, trained under different initial conditions, could be combined in an additional neural network to build an ensemble classifier, which would significantly reduce the recognition error [13]. The last provision can be illustrated with data from Table 1, giving the number of errors for each digit for the three perceptron training options corresponding to curves 4–6 in Fig. 7, after training over 660 pochs. The total number of images in the test set is 10,000.

Table 1

The number of recognition errors for each digit for three neural network training options

Digit in an image	Error quantity		
	Option 1	Option 2	Option 3
0	7	4	9
1	6	4	5
2	25	16	16
3	16	18	18
4	15	18	8
5	14	12	21
6	11	17	17
7	16	27	24
8	18	25	11
9	29	17	24

Table 1 shows that even if the images that have been misidentified are matched, combining the results of the three variants would reduce the error. And if some of the images that are misidentified by the different variants do not match up, the final error would be even smaller.

It should be noted that the frequency of pruning $L=1$ (after each epoch), as well as the magnitudes of thresholds when pruning, were chosen based on our practical experience. The issues related to the better justified and dynamic choice of a pruning frequency and the pruning thresholds magnitudes are the areas for further research. The same applies to the parameters of the procedure of distortion of input images, as well as the size of the training step.

One should note that the effectiveness of the proposed approach strongly depends on the choice of a set of pre-processing operations, which should be adequate for the practical task to be solved. All possible deviations from the samples submitted for training should be taken into consideration.

7. Conclusions

1. Pruning the connections while training a multilayer perceptron makes it possible not only to reduce the required volume of computations when using a trained network but also to improve the likelihood of the correct handwritten digit recognition. In particular, the best option to train a perceptron with pruning at the fewer connections, by 8.4 %, yielded a decrease in error from 1.89 % to 1.81 % (a gain of 1.04 times), compared to non-pruning training options.

2. The pre-processing of source data submitted to the input of a multilayer perceptron during training ensures that the learning process is regularized; it reduced the output

error from 1.89 % to 1.52 % (a gain of 1.24 times), compared to using the data unchanged.

3. The combined application of pre-distortions and pruning provides a cumulative effect that exceeds the sum of effects from the individual use of each. In particular, the margin of error was reduced from 1.89 % to 1.22 % (a gain of 1.55 times, $1.55>1.04\cdot1.24$). In addition, it has been shown that the use of predistortions in pruning regularizes the process of training a neural network and prevents retraining, which gives the possibility of monotonous reduction of connections in the perceptron at consistently high-quality recognition of handwritten digits.

References

1. Nikolenko, S., Kadurin, A., Arhangel'skaya, E. (2018). Glubokoe obuchenie. Sankt-Peterburg: Piter, 480.
2. Denil, M., Shakibi, B., Dinh, L., Ranzato, M. A., De Freitas, N. (2014). Predicting Parameters in Deep Learning. ArXiv. Available at: https://arxiv.org/pdf/1306.0543v2.pdf
3. Han, S., Pool, J., Tran, J., Dally, W. J. (2015). Learning both Weights and Connections for Efficient Neural Networks. ArXiv. Available at: https://arxiv.org/pdf/1506.02626v3.pdf
4. Cun, Y. L., Denker, J. S., Solla, S. A. (1990). Optimal Brain Damage. NIPS. Available at: http://yann.lecun.com/exdb/publis/pdf/lecun-90b.pdf
5. Denton, E. L., Zaremba, W., Bruna, J., LeCun, Y., Fergus, R. (2014). Exploiting linear structure within convolutional networks for efficient evaluation. In NIPS, 1269–1277.
6. Sainath, T. N., Kingsbury, B., Sindhwani, V., Arisoy, E., Ramabhadran, B. (2013). Low-rank matrix factorization for Deep Neural Network training with high-dimensional output targets. 2013 IEEE International Conference on Acoustics, Speech and Signal Processing. doi: https://doi.org/10.1109/icassp.2013.6638949
7. Molchanov, D., Ashukha, A., Vetrov, D. (2017). Variational dropout sparsifies deep neural networks. arXiv. Available at: https://arxiv.org/pdf/1701.05369.pdf
8. Han, S., Mao, H., Dally, W. J. (2016). Deep compression: compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv. Available at: https://arxiv.org/pdf/1510.00149.pdf
9. Qiu, J., Song, S., Wang, Y., Yang, H., Wang, J., Yao, S. 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 – FPGA'16. doi: https://doi.org/10.1145/2847263.2847265
10. Alford, S., Robinett, R., Milechin, L., Kepner, J. (2019). Training Behavior of Sparse Neural Network Topologies. 2019 IEEE High Performance Extreme Computing Conference (HPEC). doi: https://doi.org/10.1109/hpec.2019.8916385
11. Lee, N., Ajanthan, T., Torr, P. H. S. (2019). SNIP: Single-Shot Network Pruning Based on Connection Sensitivity. International Conference on Learning Representations (ICLR 2019).
12. Li, Y., Zhao, W., Shang, L. (2019). Really should we pruning after model be totally trained? Pruning based on a small amount of training. arXiv. Available at: https://arxiv.org/pdf/1901.08455v1.pdf
13. Loquercio, A., Torre, F. D., Buscema, M. (2017). Computational Eco-Systems for Handwritten Digits Recognition. arXiv. Available at: https://arxiv.org/pdf/1703.01872v1.pdf
14. LeCun, Y., Cortes, C., Burges, C. J. C. The MNIST Database of Handwritten Digits. Available at: http://yann.lecun.com/exdb/mnist/
15. 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. doi: https://doi.org/10.2991/ijcis.2017.10.1.38
16. 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: https://doi.org/10.1162/neco_a_00052
17. 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. doi: https://doi.org/10.1109/icdar.2003.1227801
18. Tarik, R. (2017). Sozdaem neyronnuyu set'. Sankt-Peterburg: OOO «Al'fa-kniga», 272.