UDC 658.62.018

A. Stenin, V. Pasko, M. Soldatova, I. Drozdovich

RECOGNITION OF HANDWRITTEN NUMBERS BASED ON CONVOLUTIONAL NEURAL NETWORKS

Abstract: This article deals with the problem of practical implementation of handwritten digit recognition based on convolutional neural networks (CNN). The CNN architecture is presented and analyzed, for which it is recommended to use cross entropy as a loss function during training, and the Softmax function as an activation function of the last CNN layer. It is also recommended to use the well-known error back propagation algorithm to implement the CNN learning algorithm. To do this, the article presents the main relations for errors at each layer.

*Keywords***:** artificial intelligence, convolutional neural network, handwritten digit recognition, MNIST database, architecture of convolutional neural network, learning algorithm

Introduction

Deep neural networks are currently becoming one of the most popular machine learning methods in the field of artificial intelligence. They show better results compared to alternative methods in such areas as speech recognition, natural language processing, computer vision [1], medical informatics [2], etc. One of the reasons for the successful use of deep neural networks is that the network automatically selects important features from the data that are necessary for solving the problem. In alternative machine learning algorithms the people distribute this functions. For this purpose, there is a specialized field of research — functional engineering. However, when processing large amounts of data, the neural network copes with the selection of features much better than a person.

The model of artificial neural networks was proposed in 1943 [4], and the term «deep learning» has been widely used in the scientific community since 2006 [5,6]. Prior to this, the terms loading deep networks [7,8] and learning deep memories [9] were used.

The growth in the popularity of deep neural networks that has been taking place in the last few years can be explained by three factors:

First, there was a significant increase in the performance of computers, including GPU (Graphics Processing Unit) computing accelerators, which made it possible to train deep neural networks much faster and with higher accuracy [10].

1. Secondly, a large amount of data has been accumulated that is necessary for training deep neural networks.

2. Third, methods of training neural networks have been developed that allow for fast and high-quality training of networks consisting of one hundred or more layers [11], which was previously impossible due to the problem of disappearing gradient and retraining.

3. The combination of three factors led to significant progress in the training of deep neural networks and their practical use, which allowed deep neural networks to take a leading position among machine learning methods.

Problem statement

One of the urgent tasks of pattern recognition, which have an undoubted practical value in many areas of human activity, is the identification of handwritten digits as objects of the classification task.

The statement of the classification problem is as follows [7]: there are many objects divided in some way into classes. A finite set of objects is given for which it is known which classes they belong to. This set is called a training sample. The class affiliation of the remaining objects is unknown. It is required to build an algorithm that can classify an arbitrary object, in our case, a handwritten digit, from the original set.

Solving the problem

As a data set, it is proposed to use a sample of the MNIST («Modified National Institute of Standards and Technology) database of samples of handwritten digits widely known in machine learning circles [9]. In it, the training sample contains 60,000 samples of digits, and the test sample contains 10,000 samples of digits. Each sample is a two-dimensional matrix of numbers from 0 to 255 with a size of 28×28 pixels. Examples of figures from this sample are shown in Fig. 1.

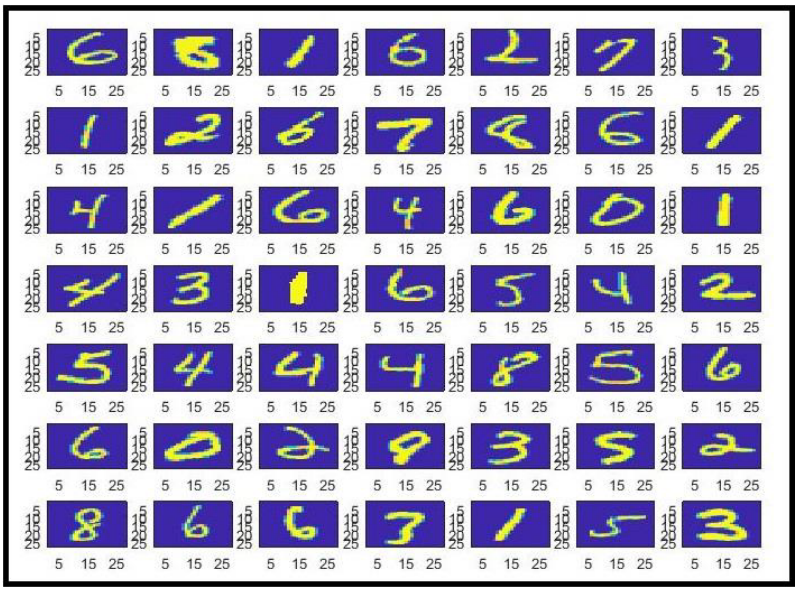

Figure 1. Examples of numbers from the MNIST sample

ISSN 1560-8956

The MNIST sample should be normalized before training, i.e. all the values of the matrices should be divided by 256 in order to bring them into the interval [0,1]. A test sample is a set of objects that is not included in the training sample and serves for the validation of a neural network. To solve the classification problem using a neural network, it is recommended to use the cross entropy as a loss function to calculate the error between the output response of the network and the reference vector:

$$C = -\sum_{i=1}^{n} \boldsymbol{Y}_i log \boldsymbol{y}_i, \quad (1)$$

where $\boldsymbol{y}_i$ – the vector of the output values of the network, and $\boldsymbol{Y}_i$ – the vector of the network reference values for a specific input sample.

As the activation function of the last layer of the convolutional neural network (CNN), it is proposed to use the *Softmax* function [8].

The *Softmax* function and its derivative for some output vector **Z**j are calculated using the following formulas:

$$\sigma(\boldsymbol{Z}j) = \frac{e^{zj}}{\sum_i^n e^{zi}}, \quad (2)$$

$$\dot{\sigma}(\mathbf{Z}j) = \sigma(\mathbf{Z}j)\big(1 - \sigma(\mathbf{Z}j)\big). \quad (3)$$

When applying the *Softmax* function to the output signal, the values of the probabilities of the input image belonging to a particular class are obtained at the output of the CNN.

To implement this task, it is necessary to build an appropriate CNN, the architecture of which is shown in Fig. 2. As can be seen from Fig. 2, the CNN consists of the following types of layers (from left to right):

1. Convolutional layer – this layer convolutes the input matrix with the convolution core. The number of convolution cores determines the number of feature maps – the first is equal to the second.

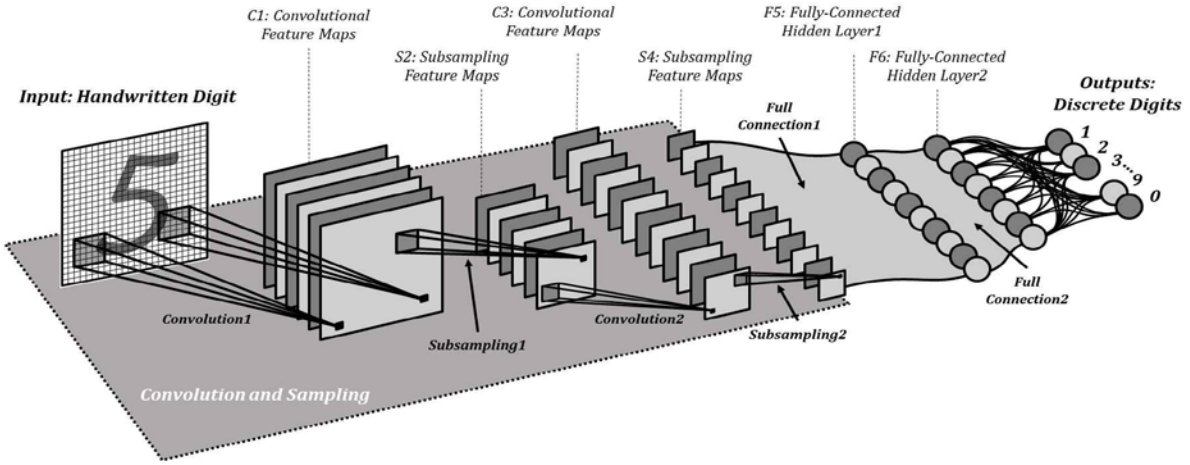

Figure 2. CNN architecture for digit classification problems

2. Subsampling, or pooling layer – this layer takes the result of the convolution of the previous layer in the form of a matrix and compresses this matrix. This is done in order to

highlight low-level features and reduce the data size. As a compression function, the arithmetic mean of the elements over the window or the maximum value over the window is most often used.

3. Fully-connected layer. A one-dimensional vector is fed to this layer from the convolutional / subsample layer standing in front of it, and this vector is obtained from the matrix by writing its elements line by line in one line.

The following main features of the CNN architecture can be distinguished for the tasks of classifying objects in the image:

– the input layer of the CNN is convolutional, and the output layer is fully connected;

– convolutional and subsample layers alternate with each other, and after their alternation, completely connected layers follow (at least 1). Thus, the final part of the CNN is nothing more than a fully-connected perceptron.

In a fully-connected CNN layer, each neuron of the first layer of neurons is connected to each neuron of the second layer of neurons (i.e., according to the principle «each with each»). The values of the neurons of the second layer are calculated using the following formulas [4]:

$$l_{2j} = F(S_{2j}), \tag{4}$$

$$S_{2j} = \left(\sum_i^n l_{1j}\,\omega_{ij}\right) - T_{2j}, \tag{5}$$

where $F(S_{2j})$ – the value of the activation function from the weighted sum S_{2j}; l_{1j} and l_{2j} – the values of the *i*-th neuron of the first layer of neurons and the *j*-th neuron of the second layer, respectively; ω_{ij} – the value of the connection between the *i*-th neuron of the first layer of neurons and the *j*-th neuron of the second layer, respectively; T_{2j} – the value of the threshold (offset) of the j-th neuron of the second layer.

In [12], a variant of the CNN was proposed to solve this problem, consisting at the input of 2 convolutional layers with sizes 28×28 and 11×11, 2 subsample layers with sizes 22×22 and 8×8, one flat layer with a size of 4×4 and 3 fully connected layers with sizes 16.30 and 10, respectively. The sizes of the convolution cores are 7×7 and 4×4. The activation functions are *ReLU* and *Softmax*.

It should also be noted that in order to effectively solve this problem of classifying objects (handwritten digits), it is usually necessary to develop a specific CNN learning algorithm that depends primarily on the chosen CNN architecture. The article [13] provides a fairly complete overview of the current state of deep neural network training methods. For CNN training, it is proposed to use a modified version of the well-known error back propagation algorithm, which refers to the methods of teaching with a teacher.

Here are the main relations for calculating the error on each of the CNN layers.

Training of a fully-connected CNN layer. The error is formed on the last layer of CNN neurons and is defined as the difference between the output response of the network (the

ISSN 1560-8956

values of the neurons of the last layer of neurons) y_i and the standard t_i [4]:

$$\gamma_i = y_i - t_i. \quad (6)$$

Next, the values of the weights and thresholds are changed according to the following formulas [4]:

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \alpha\gamma_j F(S_j), \quad (7)$$

$$Tj(t+1) = Tj(t) + \alpha\gamma j F(sj), \quad (8)$$

where α – network learning rate; *t* и *t+1* – the time points before and after the change of weights and thresholds, respectively; the indexes *i* and *j* denote the neurons of the first and second layer of neurons, respectively.

The error for a hidden layer with index i is calculated through the errors of the next layer with index j as follows [4]:

$$\gamma j = \sum_J^K F(S_j)\omega_{ij} \quad (9)$$

Fully-connected layers are trained according to the Rosenblatt training procedure, according to which the value of the learning rate is constant during the entire training time and takes values in the interval (0;1] [5]. Before hitting the convolutional or pooling layer, the one-dimensional signal is converted to two-dimensional.

Training of the convolutional and subsample layers of the CNN. The reverse propagation of the error over the subsample layer depends on the pooling function. If the pooling function is the average, then the error is evenly distributed over the m×n neurons of the block of the previous layer, and it must be multiplied by $1/(m \times n)$. That is, for each of the $(m \times n)$ neurons of the block, the error value is the same. If the pooling function is the minimum, then the error is assigned to the neuron of the block from which the maximum value for the block was taken.

The convolution operation serves as the basis for the reverse propagation of the error over the convolutional layer. When transmitting the error matrix from the pooling layer to the convolutional layer, the error matrix of the pooling layer is inversely convoluted with the matrix core.

Conclusion

It should be noted that modern architectures of convolutional neural networks are very resource-intensive, which limits the possibilities of their wide practical application. In this regard, recently there have been works that propose the architecture of the SNA, divided into hardware and software parts to increase the performance of computing. In particular, in [14], modular arithmetic was used to implement the convolutional layer of a neural network in the hardware in order to reduce resource costs.

REFERENCES

1. LeCun Y., Bengio Y., Hinton G. Deep Learning. Nature. 2015.Vol. 521. P. 436-444. DOI: 10.1038/nature14539.

2. Rav`ı D., Wong Ch., Deligianni F., et al. Deep Learning for Health Informatics. IEEE Journal of Biomedical and Health Informatics. 2017. Vol. 21, No 1. P. 4-21. DOI: 10.1109/JBHI.2016.2636665.

3. Schmidhuber J. Deep Learning in Neural Networks: an Overview. Neural Networks. 2015. Vol. 1. P. 85-117. DOI: 10.1016/j.neunet.2014.09.003.

4. McCulloch W.S., Pitts W. A Logical Calculus of the Ideas Immanent in Nervous Activity. The Bulletin of Mathematical Biophysics. 1943. Vol. 5, №. 4. P. 115–133. DOI: 10.1007/BF02478259.

5. Hinton G., Salakhutdinov R. Reducing the Dimensionality of Data with Neural Networks. Science. 2006. Vol. 313, №. 5786. P. 504–507. DOI: 10.1126/science.1127647.

6. Hinton G.E., Osindero S., Teh Y.-W. A Fast Learning Algorithm for Deep Belief Nets. Neural Computing. 2006. Vol. 18, №. 7. P. 1527–1554. DOI: 10.1162/neco.2006.18.7.1527.

7. S’ıma J. Loading Deep Networks Is Hard. Neural Computation. 1994. Vol. 6, №. 5. -P. 842–850. DOI: 10.1162/neco.1994.6.5.842.

8. Windisch D. Loading Deep Networks Is Hard: The Pyramidal Case. Neural Computation. 2005. Vol. 17, №. 2. P. 487–502. DOI: 10.1162/0899766053011519.

9. Gomez F.J., Schmidhuber J. Co-Evolving Recurrent Neurons Learn Deep Memory POMDPs. Proc. of the 2005 Conference on Genetic and Evolutionary Computation (GECCO) (Washington, DC, USA, June 25–29, 2005), 2005. P. 491–498. DOI: 10.1145/1068009.1068092.

10. Ciresan D.C., Meier U., Gambardella L.M., Schmidhuber J. Deep, Big, Simple Neural Nets for Handwritten Digit Recognition. Neural Computation. 2010. Vol. 22, №. 12.т P. 3207–3220. DOI: 10.1162/NECO_a_00052.

11. He K., Zhang X., Ren S., et al. Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (Las Vegas, NV, USA, 27–30 June 2016), 2016. P. 770–778. DOI: 10.1109/CVPR.2016.90.

12. Bredikhin A. I. Training algorithms for convolutional neural. Bulletin of the Ugra State University. -Khanty-Mansiysk: Issue 1 (52), 2019. P. 41–54. DOI: 10.17816/byusu20190141-54

13. Sozykin A.V. Review of training methods for deep neural networks. Bulletin of SUSU. Series: Computational Mathematics and Computer Science. Vol. 6, №. 3. 2017. - P. 28-59. DOI: 10.14529/cmse170303.

14. Chervyakov N.I., Lyakhov P.A., Nagornov N.N., Valueva M.V., Valuev G.V. Hardware implementation of a convolutional neural network with the use of calculations in the system of residual classes. Computer Optics. 2019. Vol. 43, №. 5. P. 857-868. DOI: 10.18287/2412-6179-2019-43-5-857-868.

ISSN 1560-8956