Recognition of the MNIST dataset with defective rows

A N Shevlyakov 1, A A Berezin 2

1 Sobolev Institute of Mathematics, 13 Pevtsova str, Omsk 644099, Russia
2 Omsk State University, 55 Mira str. Omsk 644077, Russia
E-mail: art.shevlyakov@gmail.com

Abstract. One of the well-known classification problems in machine learning is the problem of recognizing handwritten numbers. This task is solved, since there are different neural networks that determine with fairly high accuracy which number is shown in the picture. In this paper, we will consider the recognition of handwritten digits subjected to certain deformations.

1. Introduction
The problem of recognizing handwritten numbers is an example of a machine learning problem "with a teacher", that is, we will receive data with the correct answers at the input. The problem is that there are 10 classes of objects - numbers from 0 to 9, and it is necessary to refer the image that comes to the input of the neural network to one of these classes.

2. Data Description
Samples of handwritten numbers from MNIST will be used as the main data for the study in this work. MNIST is a voluminous database that contains various examples of handwritten numbers [2]. This database is represented by 70,000 different copies of handwritten numbers (Fig. 1): 60,000 pictures for training and 10,000 for testing a neural network. The dataset is in the public domain [3].

Each digit is centered and normalized to size in each individual image. The MNIST database contains a file with images and a file with class labels - the correct answers. Images in grayscale (0 - white, 255 - black) with a size of 28 × 28 pixels are recorded in binary form in one file. The background of the image is white and the numbers are painted in dark colors. Recording in binary form is considered convenient from the point of view of processing all images by a neural network, but not entirely convenient for human perception.

Figure 1. Examples of numbers from the MNIST library
The use of the MNIST database is beneficial in that to check the operation of the neural network, you do not need to pre-process the data, and it contains a very large number of sample numbers, sufficient to check the quality of training the neural network.

3. Model Description
To recognize numbers from the MNIST set, a deep neural network was built, which consists of 8 layers: four convolutional layers, two subsampling layers, and two more fully connected layers. In fig.2 shows how many channels are obtained as a result of convolutional layers and how the dimensions of the images change.

4. Deformation of numbers
4.1. Describing the deformation of handwritten numbers
In this work, the following deformation of images will be presented: on each image of $28 \times 28$ pixels, horizontal rows will be cut out in a pixel manner. An example of such deformation is shown in the following figure.
In fig. 3 on the left shows the correct number "2", which was obtained by cutting out rows numbered 4, 5, 13, 15 and 23.

In fig. 4 shows the number "7" before and after removing three rows: 9, 17, 25.

In fig. 5 shows the deformation of a unit after deleting lines numbered: 0, 1, 2, 10, 11, 12, 16, 18, 19, 24.

You can see that in all three figures, the rows were chosen at random, so that even those lines that are not part of a handwritten number, that is, white lines that did not have numbers, could be generated. For example, these are rows numbered 0, 1 or 26 - they are completely painted over in white in almost all pictures.
4.2. The results of the neural network on damaged images

To check the operation of the network on damaged images, we will test the network, starting with cutting from 0 to 27 rows. To do this, in the loop, in order, we will: first initialize the test sample, then deform it and feed it to the input to the method for performing NN testing, which outputs the results.

As a result, we got the fact that the neural network works and recognizes all damaged images, but the only question is how it does it and how well it does it. In fig. 6 shows a graph of the accuracy of the network, depending on how many rows were cut. You can see that the accuracy graph monotonically decreases from 99.41% to ~ 10%.

![Graph of accuracy on damaged digits](image)

**Figure 6. Accuracy on damaged digits**

Based on the graph, we can conclude: when cutting one or three rows, the network shows a pretty good result of about 96%. Then it behaves unstable: when the number of deleted rows is equal to 24-27 out of 28, the network starts to "guess" - from the theory of probability, the chances of guessing one of ten digits are 0.1, so the accuracy obtained at the last iterations is about 10 percent.

It will also be interesting to look at the graph of the loss function (Fig. 7). Of course, the error grows rapidly, because the data deteriorates and it is difficult for neural networks to identify any common signs in the pictures, since we randomly paint over the rows, and the convolutional neural network takes into account the contiguity of pixels. It can be noted that on rows 25-27 the value of the loss function began to decrease - this is due to random reasons and is not statistically significant. In particular, this may be due to the fact that the input weights are initialized with a normal distribution.
From this data, the critical value of the cut rows was determined for the model, where the accuracy drops most strongly compared to the previous value - this number is 9 rows. The difference in precision for this number of rows with the previous value is 6.57%.

As it turned out, the network can recognize numbers, but with different accuracy. For example, in Figure 8 shows images in which 9 random rows of pixels were removed, and the neural network correctly determined which classes the image data belongs to: 3, 7, 4, 5, 6, 2, accordingly.

5. Conclusion
This work was devoted to the study of deep convolutional neural networks using the example of the problem of recognizing handwritten digits with damaged parts of the image. It has been found that it is possible to classify damaged numbers, but with varying accuracy. A satisfactory result of about 96% was obtained for images with distortion from one to three rows.
The program code was written in Google Colab in Python. Access mode: https://github.com/Andrey-Berezin/neural_networks, free. The components of the Keras library were used to create neural network models. The MNIST database was used as data for working with NS. Accuracy and error plots were plotted using the Matplotlib library.

Acknowledgments
The research of the first author was funded in accordance with the state task of the IM SB RAS, project FWNF-2022-0003.

References
[1] Nikolenko S, Kadurin A, Arkhangeskaya E 2018 Deep learning (SPb.: Peter) p 480