

Convolutional neural networks for image classification

Andrii O. Tarasenko, Yuriy V. Yakimov, Vladimir N. Soloviev^[000-0002-4945-202X]

Kryvyi Rih State Pedagogical University, 54, Gagarina Ave, Kryvyi Rih 50086, Ukraine
{vnsoloviev2016, urka226622, andrejtarasenko97}@gmail.com

Abstract. This paper shows the theoretical basis for the creation of convolutional neural networks for image classification and their application in practice. To achieve the goal, the main types of neural networks were considered, starting from the structure of a simple neuron to the convolutional multilayer network necessary for the solution of this problem. It shows the stages of the structure of training data, the training cycle of the network, as well as calculations of errors in recognition at the stage of training and verification. At the end of the work the results of network training, calculation of recognition error and training accuracy are presented.

Keywords: machine learning, deep learning, neural network, recognition, convolutional neural network, artificial intelligence.

1 Introduction

Today, a very old field of research, called artificial intelligence deals with the ability of machines to think like a man. Leading it companies and researchers from universities have made a breakthrough in the research of artificial intelligence and now we have software that can “see” (recognize objects in the image, capable of restoring video), make predictions based on certain data (forecasting stock market indices), make decisions (the ability to play games, sometimes better than a person).

The idea of artificial intelligence originated in the 1940s, when the question first arose whether it was possible to make a computer “think” [10]. Briefly, this sphere can be described as follows: automation of intellectual tasks, which are usually performed by people [1]. After some time, the researchers were able to create models that are capable of learning and perform tasks without a clear statement. Such models are called neural networks. Their peculiarity lies in the ability to learn without programming the networks themselves.

Artificial intelligence has occupied the place of the sphere of research and today includes the paradigm of learning – machine learning [12; 13], which differs from the initial, as it was previously called “symbolic artificial intelligence”, in that in machine learning, the programmer enters into the program data with answers that correspond to these data, and the output receives the rules (the answer), and symbolic artificial intelligence, in turn, performed the rules set by the programmer.

In recent years, much attention has been paid to deep learning, which is successfully used in classification and recognition problems. The key place here is occupied by

neural networks, namely convolutional neural networks, which contain the meaning of “depth” [2].

Depth in deep learning does not mean the deeper understanding achieved by this approach, the idea is multi-layered representation. The number of layers into which the data model is divided is called the depth of the model. Other relevant names for this area of machine learning could be: layered learning or hierarchical learning.

Modern deep learning often involves tens or even hundreds of successive layers of representation – all of which are determined automatically by the training data.

2 Analysis of previous studies

Deep neural networks over the past 20 years take a very large part of the world’s research and development, but the first mention of deep network algorithms appeared in the mid-1960s in the book of Aleksey G. Ivakhnenko and Valentin G. Lapa [6]. Then, the concept of “deep learning” in the scientific community emerged through the work of Rina Dechter in 1986 [3].

The first models of convolutional neural networks were called “neocognitron” and were discovered in 1980 by Kunihiko Fukushima [4]. Fukushima proposed several algorithms for supervised and unsupervised learning, and the neocognitron itself was a multilayered deep structure.

Taking into account the fact that the recognition of medical images has recently attracted the attention of researchers, there are several difficulties in the study of this area, namely: 1) a small number of training copies, 2) the difference in scale and fuzzy boundaries of images. These disadvantages were taken into account when creating a network model that offers a full-scale convolutional layer extracting patterns of different receptive fields with a common set of convolutional nuclei, so that scale-invariant patterns are captured by this compact set of nuclei [13].

Last decade convolutional networks are gaining a lot of attention of researchers and developers. ImageNet is one of the largest competitions dedicated to artificial intelligence and computer vision. Among the varieties of artificial networks, the prizes were taken using a convolutional network structure, such as AlexNet [7], VGG [14], GoogleNet [15] and ResNet [5]. These networks do an excellent job and have a large percentage of recognition of more than 90%.

New developments in the recognition of biomedical images Shuchao Pang, Anan Du, Mehmet A. Orgun and Zhezhou Yu [9]. This new neural network, which is called “fused” convolutional neural network (FCNN) has a precise and highly efficient classifier, which combines the features of small balls and features of deep layers.

3 Basic information

Neurons transmit electrical impulses. When transmitting a pulse (through an axon), the signal can be amplified or attenuated to be transmitted to a trace neuron (through dendrites). Such basic functions are implemented in the model of artificial neural networks:

1. Data. As signals, the artificial network uses input data (photos, audio files), which are reduced to a certain form, which the network is able to read;
2. Weights are the parameters of each layer of neurons. Act as a force signals by analogy with natural neurons. If the natural neurons the strength of the impulses, the artificial network is a numeric value;
3. Next comes the input function. In this part, the data and numerical values of the weights are processed according to the type of convolution layer;
4. This is followed by the activation function. The function that processes the input data, its value, is the output of the neuron. Then the data are transferred through the scales to the next layer of neurons (Fig. 1).

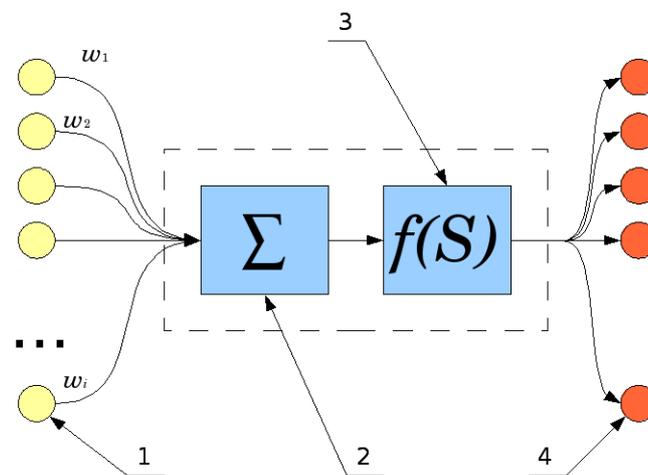


Fig. 1. Representation of artificial neuron [20].

As mentioned earlier, an artificial neural network consists of a large number of artificial neurons, which are combined into layers forming a network of neurons. There are the following types of layers of neurons:

- Input layer. Here the network receives input data that is converted in advance to the desired format (lists, arrays);
- Output layer. A layer that processes the incoming data in the outgoing (arrays of numbers) that satisfy the task;
- Convolutional layers. Models of deep neural networks use convolutional layers, which perform various kinds of “learning” operations of input data transferring to the output layer.

The simplest models of neural networks do not use convolutional layers, they are also called “single-layer” (Fig. 2). Input data described above – (x_1, \dots, x_n) , relevant learning factors – (w_{nm}) , activation function – (Σ) , and actually the output data – (y_1, \dots, y_m) .

Deep learning uses convolutional neural networks, which can contain many convolutional layers of neurons. The task of such layers is to process the input data in

such a way that the output receives data that satisfies the problem condition. As with a single-layer network, these are lists of numbers.

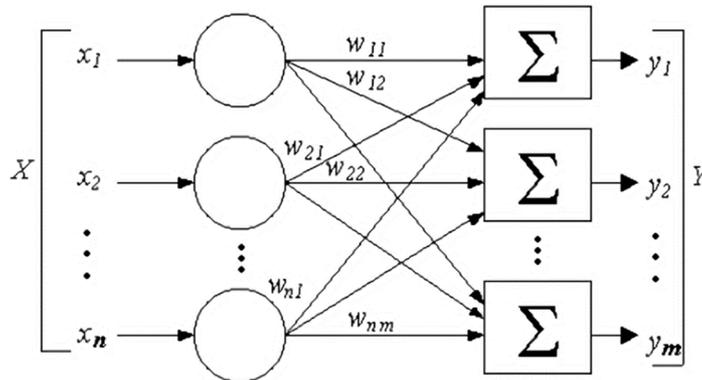


Fig. 2. Model of single-layer neural network.

Deep learning uses the concept of a container tensor for data. In fact, the tensor is a matrix of numbers, or one number is also called a tensor, or else – a scalar.

Tensors are characterized by the following key characteristics:

- Number of axes (rank, dimension). For example, a matrix is a two-dimensional tensor;
- Form. A list of integers describing the number of dimensions on each axis of the tensor;
- Data type. The type of data belonging to the tensor; for example: float32, float64, uint8, etc.

It should also be noted that, although the data is stored in the tensor as a mass, the neural network does not process the entire data set at once, but divides it into so-called “packets” of data. That is, the data package represents several separate samples with their other characteristics.

There are the following main categories of data:

- Vector-two-dimensional tensors with shape (samples, features);
- Time series-three-dimensional tensors with shape (samples, time labels, signs);
- Image-four-dimensional tensors with shape (samples, height, width, color);
- Video-five-dimensional tensors with shape (samples, frames, height, width, color).

Also, returning to the structure of the neuron, it is necessary to mention the function of action. It has been said that the function takes some value as a parameter. This value is the data that has gone through operations on the tensors (each convolutional layer performs its own operations), after which the data becomes the source of this convolutional layer.

Depending on the complexity of the task set for the neural network, activation functions can be different (table 1):

Table 1. Some of the graphs and descriptions of some activation functions.

Figure	Function name	Description
	Unit function $f(x) = \begin{cases} 1, & x \leq b \\ 0, & x \geq b \end{cases}$ b – adder output.	As you can see in the graph, the function has two states 0 or 1. It is usually used in tasks where it is enough to give the answer “Yes” or “No”
	Logistic function $f(x) = \frac{1}{1 + \exp(-a*x)}$ a – the coefficient that characterizes the curvature of the graph	The most commonly used function is due to the fact that among the two States 0 and 1 there are many others, for example 0.234532, or 0.7. This makes it possible to get from the neural network not only one answer, but, for example, 10 or 1000
	The hyperbolic tangent $f(x) = \tan \frac{x}{a}$	It is used for a more realistic model of a neural network, which can give the initial values not only positive, but also negative, that is, from -1 to 1

The following are the steps that are performed in the training cycle:

- The neural network receives a data packet with training instances and corresponding data for verification (must be different);
- The network performs data processing (this step is called a direct pass) and receives a packet of predictions;
- An estimate of the discrepancy between the network prediction and the validation data is calculated, that is, there must be a function to estimate this discrepancy;
- The parameters are adjusted to reduce discrepancies on this data packet.

The learning cycle is repeated as many times as the problem condition requires. After completing the training, we will get a network that has a low grade of disagreement.

Correction parameters occurs in the calculation of the gradient differences of network parameters. In this case, an offset parameter is added to the training parameters, which is the opposite of the network variance gradient.

Then, the training cycle will look like this:

1. The neural network receives a data packet with training instances and corresponding data for verification (must be different);
2. The network performs data processing (this step is called a direct pass) and receives a packet of predictions;

3. An estimate of the discrepancy between the network prediction and the validation data is calculated, that is, there must be a function to estimate this discrepancy;
4. The variance gradient for the network parameters (reverse) is calculated);
5. The parameters are adjusted by a small value in the direction opposite to the gradient to reduce discrepancies on this data packet.

To achieve the result, apply the gradient descent method to the gradient of differences on the selected data package. Before the training cycle, the point of calculating the loss gradient on a certain data package is entered, after which the global minimum of this function is calculated by the gradient descent method.

Fig. 3 images of the gradient descent operation on the function are presented:

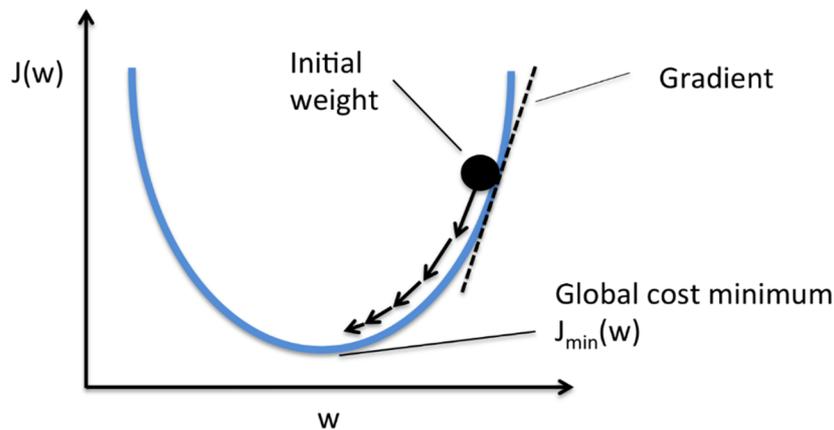


Fig. 3. Gradient descent.

The architecture of neural networks is determined by the tasks for which neural networks are designed.

The task of this work is the recognition of objects in images. Therefore, it is necessary to use the appropriate network architecture, which highlights the features in the images and forms a representation of them about the object in the photo.

We review the principle of convolutional neural networks for image distribution.

First, you need to determine the type of architecture. As described earlier, convolutional neural networks in most cases have a sequential architecture, where the neural network receives certain data at the input (tensors), after which the data is sequentially processed by the source layer-sequential architecture.

Convolutional layers are used for recognition, in which the features of each object are selected. For example, the image falls on the input layer, then the neural network tries to select one common image (usually in simple networks to recognize one object). Selection of an object is performed by means of so-called layer filters. The filter is a small window relative to the image that reads a certain area of the image. Usually the window size is 3x3 pixels, or 5x5 pixels, with different image sizes. In order to determine the image values, such a filter must pass through the image. Usually the filter starts to recognize from the upper left corner of the image and moving 1 pixel to the

side, and then to the bottom passes through the entire image. At the initial stages of research, it is advisable to use a 28x28 pixel image (Fig. 4). Because large images require more time passage, and therefore more time to learn.

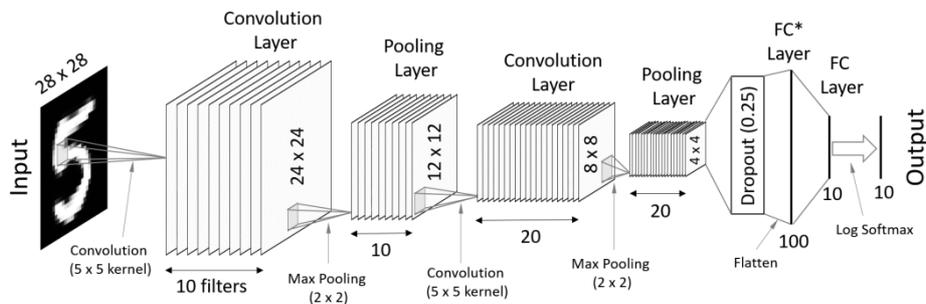


Fig. 4. Convolutional neural network architecture [16].

So, at the first convolutional stage, the neural network uses filters to determine the main object in the image.

The next step in network training is to highlight the spatial hierarchy of features. That is, having identified in the picture, for example, a cat, the network must divide the cat into parts of objects, which also must “remember” (Fig. 5). After training, the network will form a representation of such objects about the object as a whole, that is, about a cat, or other object of recognition.

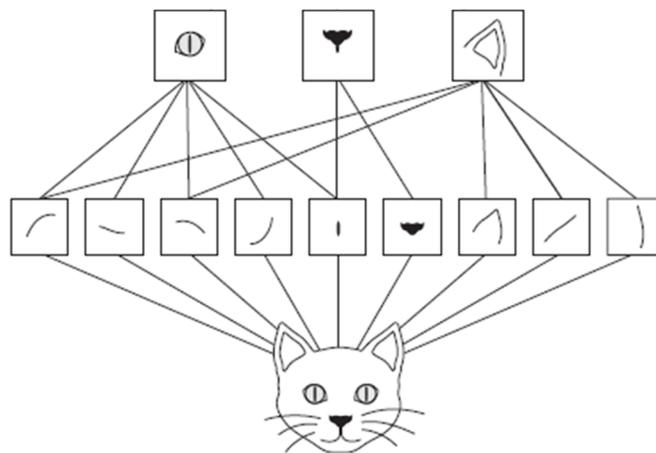


Fig. 5. Define the spatial hierarchy of features.

So, we have determined that using convolutional layers of filters with a certain size, to determine the primary features of the object in the image.

However, with the help of convolutional layers alone, it is impossible to achieve recognition of the spatial hierarchy of objects. To achieve this goal, you should perform

certain operations on the image. One of these operations, which is most often used in practice – “selection of the largest of the neighboring” (Max Pooling).

The reason for using Max Pooling is that the neural network must determine the spatial hierarchy of objects, and for this we need to reduce the image by 2 times.

The Max Pooling operation is similar to the convolution operation. Take the window, now 2x2 pixels and perform one simple action with the taken four elements-Tami-choose the largest number. Thus, the initial data is filled only with the largest numbers for each window and the resulting image is reduced by half.

The third step is to define a data set for neural network training.

Since classification tasks are primarily supervised learning (teacher-assisted learning), then we will use a special data package that contains training data and feature class labels that refer to the recognized data.

The sigmoid activation function is suitable for classification problems, but in our case we will use the ReLU activation function (Fig. 6). The advantages of using such a function are the sparsity of the activity, that is, the involvement of only a part of neurons, which will accordingly reduce the load in the calculation.

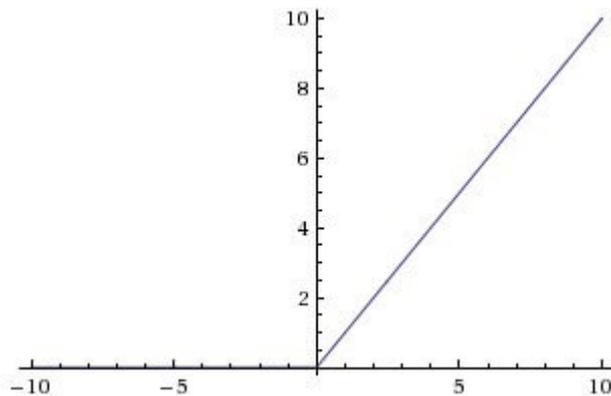


Fig. 6. Graph of ReLU function [1].

Although the use of this activation function makes part of the network passive (some neurons are not activated), it produces good results on testing [2].

$$f(x) = \max(0, x)$$

To train the network, the change of the corresponding weights for each neuron is used, namely, they are changed with the help of the optimizer. The optimizer uses gradient descent in this case.

4 Development tool

An important stage in the development of the program is the choice of development tools, because it affects the complexity and quality of the result. In fact, any

programming language has tools for the development of neural networks, the difference is only in the complexity of their implementation in the chosen language. There are even many implementations of neural networks in the web programming language. Today Python is quite a popular language for researchers. After all, it is quite easy to understand and has a large community of developers. Python has many libraries for research and data visualization, so it is advisable to choose Python as the language of implementation of your own software tool.

To control the versions of the libraries and the development environment, it is advisable to choose the Anaconda tool. The choice of Anaconda lies in the convenience of version control of programming environments and libraries. Anaconda has a base of libraries for Python, which are quite convenient to install, and supports several programming environments, one of which is Jupyter Notebook, which was originally developed for the convenience of research, but eventually gained popularity among the community of developers. It will also be used as a programming environment.

To achieve this goal in the beginning, we will use a lot of additional libraries, in particular should be allocated Keras and TensorFlow.

Keras is an open source deep learning library written in the Python programming language. The library was created to improve research and design of neural networks. The library has a lot of implementations of convolutional layers, optimizers, activation functions, work with images and text.

TensorFlow is an open-source software library for machine learning developed by Google to solve the problems of building and training a neural network in order to automatically find and classify objects, achieving the quality of human perception. It is used both for research and for the development of Google's own products. The main API for working with the library is implemented for Python.

NumPy is an open-source module for python that provides General mathematical and numerical operations in the form of pre-compiled, fast functions. They are combined into high-level packages. They provide functionality that can be compared to that of MatLab. NumPy (Numeric Python) provides the base methods for handling the large arrays and matrices. SciPy (Scientific Python), which we will also use, extends numpy functionality with a huge collection of useful algorithms such as minimization, Fourier transform, regression, and other butt-end mathematical techniques.

Matplotlib is a Python programming language library for visualization of two-dimensional (2D) graphics data (3D graphics is also supported). The resulting images can be used as illustrations in the publication.

OpenCV is a computer vision and machine learning library. We use the OpenCV library to take a picture from a webcam.

PIL (python image library) – a set of tools for working with images in Python.

Dlib is an open source machine learning library. Dlib contains in its database a pre-trained neural network that recognizes the descriptors of the person's face from the photo, use in future work.

Tkinter is a cross-platform library for building window interfaces, included in the standard set of Python modules.

We used as a data set for training and testing x-ray images of fractures of human body parts. The images are taken from such datasets:

- MURA;
- MedPix;
- OmniMedicalSearch.

The learning process is shown in Fig. 7. In the picture, each iteration of the learning (epoch) is accompanied by calculations of errors and accuracy on the training data set and on the test data set. The data to be checked contain images that are not present in the training set, so this accuracy should be guided. The number of epochs 25 for testing was randomly selected. However, analyzing the accuracy of recognition with each epoch, it becomes clear that at first the neural network increases the percentage of accuracy, and then there are fluctuations by several percent. Starting from the 8th epoch, the accuracy falls, then increases, then repeats until the end. This is a clear sign of retraining the network.

```

Epoch 1/25
- 148s - loss: 1.6839 - acc: 0.3842 - val_loss: 1.3082 - val_acc: 0.5328
Epoch 2/25
- 139s - loss: 1.3321 - acc: 0.5198 - val_loss: 1.1408 - val_acc: 0.6042
Epoch 3/25
- 142s - loss: 1.1922 - acc: 0.5769 - val_loss: 1.0420 - val_acc: 0.6432
Epoch 4/25
- 143s - loss: 1.1047 - acc: 0.6096 - val_loss: 0.9681 - val_acc: 0.6670
Epoch 5/25
- 144s - loss: 1.0282 - acc: 0.6382 - val_loss: 0.9166 - val_acc: 0.6896
Epoch 6/25
- 146s - loss: 0.9746 - acc: 0.6555 - val_loss: 0.9387 - val_acc: 0.6826
Epoch 7/25
- 147s - loss: 0.9281 - acc: 0.6721 - val_loss: 0.8726 - val_acc: 0.7032
Epoch 8/25
- 144s - loss: 0.8862 - acc: 0.6856 - val_loss: 0.9464 - val_acc: 0.6610
Epoch 9/25
- 148s - loss: 0.8496 - acc: 0.7018 - val_loss: 0.8446 - val_acc: 0.7118
Epoch 10/25
- 146s - loss: 0.8242 - acc: 0.7096 - val_loss: 0.7807 - val_acc: 0.7326
Epoch 11/25
- 146s - loss: 0.7889 - acc: 0.7216 - val_loss: 0.7913 - val_acc: 0.7246
Epoch 12/25
- 146s - loss: 0.7630 - acc: 0.7312 - val_loss: 0.8651 - val_acc: 0.6990
Epoch 13/25
- 147s - loss: 0.7261 - acc: 0.7437 - val_loss: 0.8043 - val_acc: 0.7278
Epoch 14/25
- 146s - loss: 0.7107 - acc: 0.7496 - val_loss: 0.8587 - val_acc: 0.7158
Epoch 15/25
- 145s - loss: 0.6936 - acc: 0.7576 - val_loss: 0.7686 - val_acc: 0.7368

```

Fig. 7. Plotted accuracy on the data for verification.

The plot at fig. 8 shows that the accuracy increases to the tenth epoch, then decreases, then increases. The number of learning epochs should be reduced.

Fig. 9 shows the errors on the same data. The smallest error is recorded on the tenth epoch.

During training, we obtain certain data after the completion of each era of training: loss – error on training data; acc – accuracy on training data; val_loss – error data for testing; val_acc – precision on the data for verification.

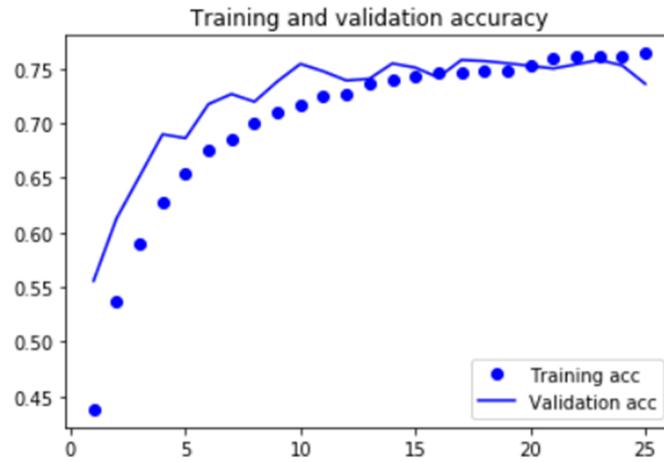


Fig. 8. Accuracy on data for training and verification.

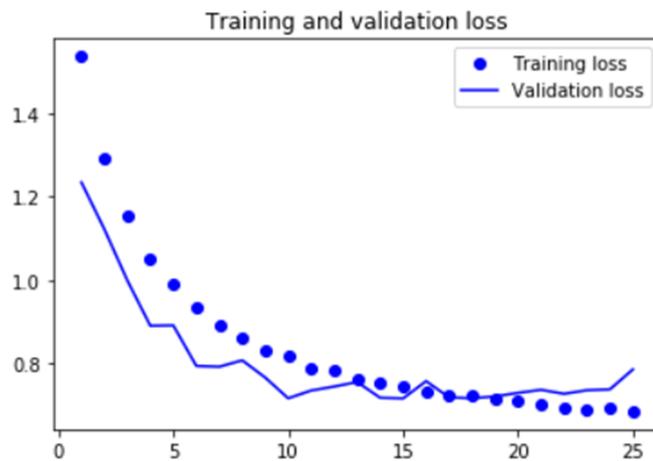


Fig. 9. Chart errors.

As a result of the training, we received an accuracy of 73.13% on the test data.

We use of Convolutional neural network in the recognition of medical images. The neural network is trained on a set of data from broken and whole human bones (Fig. 10). The neural network is able to identify obvious bone fractures on X-rays (Fig. 11).

5 Conclusion

Consequently, in this the important bases of structure and a structure of convolution neural networks and a cycle about the theory of neural networks were shown. As it was

shown in the end, the result of training of neural networks allows, without their explicit programming, “teach” the program to recognize objects in the images.

This paper provides examples of databases, both existing and self-assembled, on which the neural network learns to distinguish objects. The assessment of accuracy and errors at the stages of training and verification was also carried out.



Fig. 10. Selection of objects.

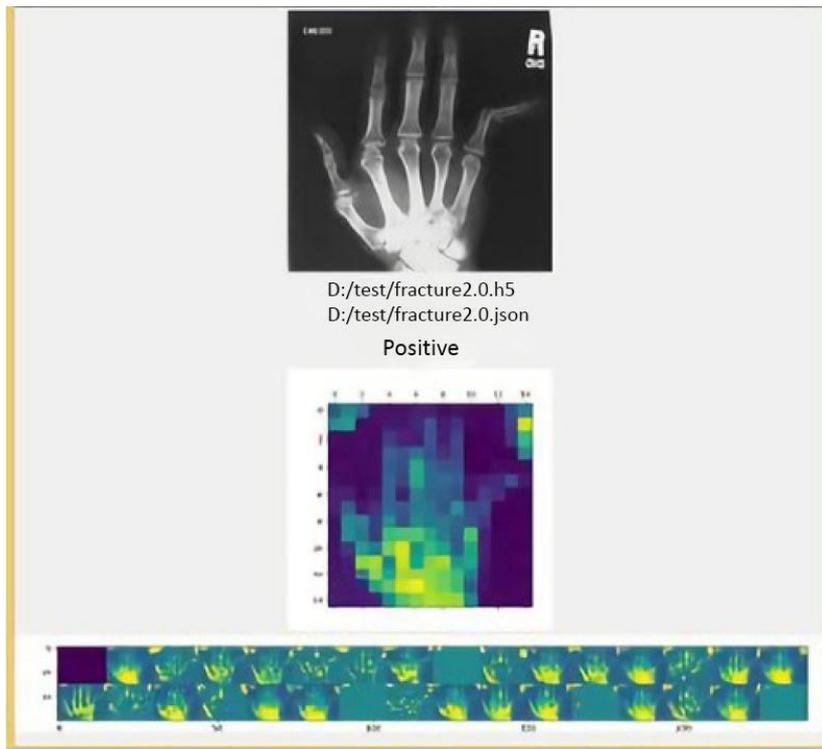


Fig. 11. Result of recognition of a fracture on a finger of the person.

References

1. Brownlee, J.: A Gentle Introduction to the Rectified Linear Unit (ReLU). Machine Learning Mastery. <https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/> (2019). Accessed 25 Oct 2019
2. Brownlee, J.: A Tour of Machine Learning Algorithms. Machine Learning Mastery. <https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/> (2019). Accessed 25 Oct 2019
1. Chollet, F.: Deep Learning with Python. Manning, Shelter Island (2017)
2. Courville, A., Goodfellow, I., Bengio, Y.: Deep Learning. MIT Press, Cambridge (2016)
3. Dechter, R.: Learning while searching in constraint-satisfaction-problems. In: AAAI-86 Proceedings The Fifth National Conference on Artificial Intelligence, August 11–15, 1986, in Philadelphia, Pennsylvania., pp. 178–183. <https://aaai.org/Papers/AAAI/1986/AAAI86-029.pdf> (1986). Accessed 17 Aug 2019
4. Fukushima, K.: Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics* **36**, 193–202 (1980). doi:10.1007/BF00344251
5. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, 27–30 June 2016, pp. 770–778. IEEE (2015). doi:10.1109/CVPR.2016.90
6. Ivakhnenko, A.G., Lapa, V.G.: Cybernetics and forecasting techniques. American Elsevier Publ. Co., New York (1967)
7. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. *Communications of the ACM* **60**(6), 84–90 (2017). doi:10.1145/3065386
8. Laves, M.H., Ihler, S., Ortmaier, T.: Deformable Medical Image Registration Using a Randomly-Initialized CNN as Regularization Prior. In: Medical Imaging with Deep Learning 2019. <https://openreview.net/pdf?id=S1ehZfQ15E> (2019). Accessed 25 Oct 2019
9. Pang, S., Du, A., Orgun, M.A., Yu, Z.: A novel fused convolutional neural network for biomedical image classification. *Medical & Biological Engineering & Computing* **57**, 107–121 (2019). doi:10.1007/s11517-018-1819-y
10. Semerikov, S.O., Teplytskyi, I.O., Yechkalo, Yu.V., Kiv, A.E.: Computer Simulation of Neural Networks Using Spreadsheets: The Dawn of the Age of Camelot. In: Kiv, A.E., Soloviev, V.N. (eds.) Proceedings of the 1st International Workshop on Augmented Reality in Education (AREdu 2018), Kryvyi Rih, Ukraine, October 2, 2018. CEUR Workshop Proceedings **2257**, 122–147. <http://ceur-ws.org/Vol-2257/paper14.pdf> (2018). Accessed 30 Nov 2018
11. Semerikov, S.O., Teplytskyi, I.O., Yechkalo, Yu.V., Kiv, A.E.: Computer Simulation of Neural Networks Using Spreadsheets: The Dawn of the Age of Camelot. In: Kiv, A.E., Soloviev, V.N. (eds.) Proceedings of the 1st International Workshop on Augmented Reality in Education (AREdu 2018), Kryvyi Rih, Ukraine, October 2, 2018. CEUR Workshop Proceedings **2257**, 122–147. <http://ceur-ws.org/Vol-2257/paper14.pdf> (2018). Accessed 30 Nov 2018
12. Semerikov, S.O., Teplytskyi, I.O.: Metodyka uvedennia osnov Machine learning u shkilnomu kursi informatyky (Methods of introducing the basics of Machine learning in the school course of informatics). In: Problems of informatization of the educational process in institutions of general secondary and higher education, Ukrainian scientific and practical conference, Kyiv, October 09, 2018, pp. 18–20. Vyd-vo NPU imeni M. P. Drahomanova, Kyiv (2018)

13. Semerikov, S.O.: Zastosuvannia metodiv mashynnoho navchannia u navchanni modeliuвання maibutnikh uchyteliv khimii (The use of machine learning methods in teaching modeling future chemistry teachers). In: Starova, T.V. (ed.) Technologies of teaching chemistry at school and university, Ukrainian Scientific and Practical Internet Conference, Kryvyi Rih, November 2018, pp. 10–19. KDPU, Kryvyi Rih (2018)
14. Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. In: International Conference on Learning Representations. <https://www.robots.ox.ac.uk/~vgg/publications/2015/Simonyan15/simonyan15.pdf> (2015). Accessed 25 Oct 2019
15. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going Deeper with Convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, 7–12 July 2015, pp. 1–9. IEEE (2015). doi:10.1109/CVPR.2015.7298594
16. Theart, R.: Getting started with PyTorch for Deep Learning (Part 3: Neural Network basics). Code to Light. <https://codetolight.wordpress.com/2017/11/29/getting-started-with-pytorch-for-deep-learning-part-3-neural-network-basics/> (2017). Accessed 25 Oct 2019