ANALYSIS OF THE BRAIN COMPUTER TOMOGRAPHY RESULTS USING THE CONVENTIONAL NEURAL NETWORK

Wang Ji
Xinjiang, People's Republic of China
wswj969408979@gmail.com

Voronov V.I.
Moscow Technical University of Communications and Informatics, Moscow, Russia
vorvi@mail.ru

DOI: 10.36724/2664-066X-2020-6-5-6-11

ABSTRACT

Advances in technology are making health research increasingly complex. Artificial intelligence is widely used in this research. Convolutional neural networks are one of the most common and optimal algorithms for working with images. Image recognition results are used to analyze the results of medical examinations of patients. The subject of the research – analysis of the human brain computed tomography results using a convolutional neural network based on the Keras library.

KEYWORDS: *Artificial intelligence, convolutional neural network, keras, convolutional layer, brain tomography.*

INTRODUCTION

Modern hospitals create a large number of medical images every day. These images are transferred to a special cloud center, which allows organizing their storage and processing. The main task in processing medical images is to extract various medical signs from them, which in the future are also accumulated and processed. Due to the significant progress made in the application of deep learning methods, some of them are successfully applied in the field of medical image analysis. One such method is a convolutional neural network (CNN).

Information about authors

Wang Ji, *graduate student, Xinjiang, People's Republic of China*
Voronov V.I., *Ph.D., Associate Professor, Moscow Technical University of Communications and Informatics, Moscow, Russia*

I. NEURAL NETWORKS

Neural networks are found used in various fields of science and technology [1,2]. If we talk about medicine, then the scope includes the analysis of images and video streams, data series, classification of signs and much more [3,4,5].

Keras

Keras is an extended open source neural network library written in Python that supports GPU and CPU as well as a highly modular library of neural networks themselves. Keras is currently used in the Theano and TensorFlow libraries. Keras provides APIs that allow users to focus on developing models and experiment with models faster.

The APIs integrate many small components from Tensorflow and Theano as modules, so a network built using those two can also be built with Keras, with no performance penalty. The main advantage of using the Keras infrastructure is that it can seriously save the developer's time when setting up the created network structure [6].

Convolutional neural network (CNN)

Convolutional Neural Network, CNN or ConvNet is a feedforward neural network [7,8]. It is usually formed from a convolutional layer, a convergence layer (activation and pooling) and a fully connected layer. Training is carried out on the basis of a backpropagation algorithm. When using a feed-forward neural network, a problem arises at the training stage related to the fact that there are too many parameters and it is difficult to extract local invariant functions. Convolutional neural networks have three structural features: local connections [9], weight distribution, and equal variation. These features make the convolutional neural network somewhat invariant to movement, scaling and rotation [10].

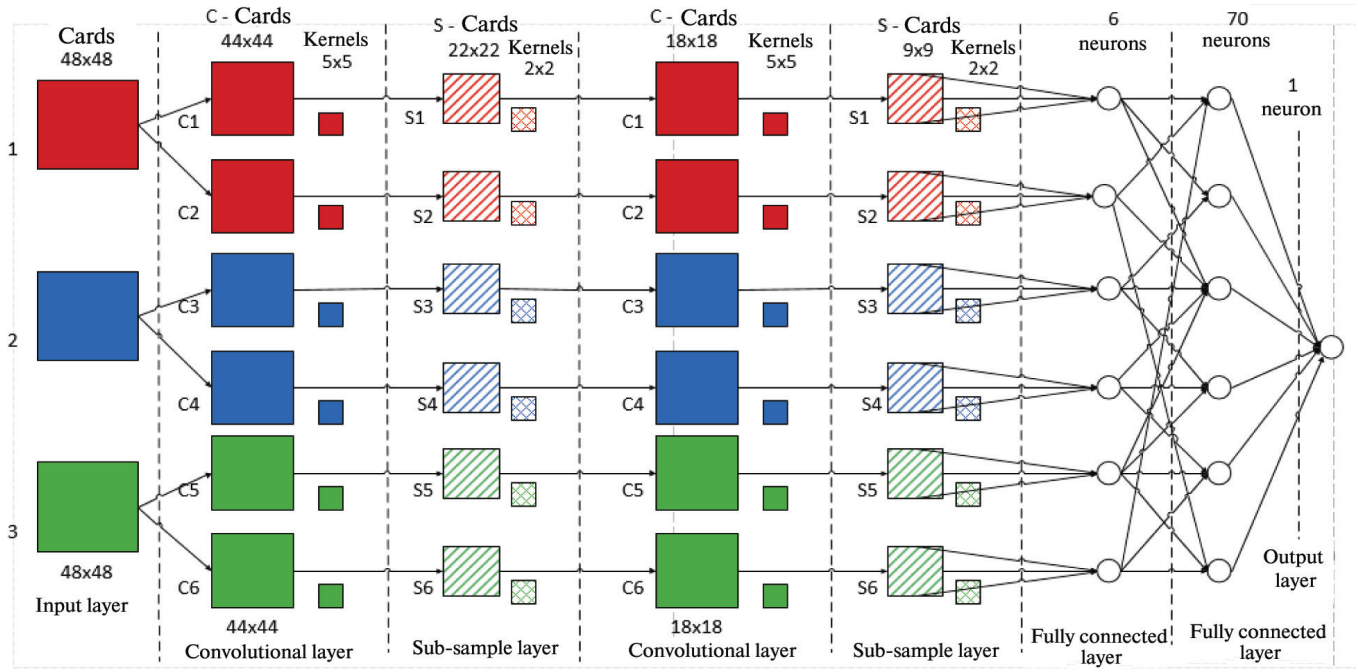

Figure 1. Convolutional neural network architecture

II. DATA SET

For work, we will use the prepared set of images created by Felipe Kitamura. These images are publicly available CT images of the brain from Google Images [11]:

https://www.kaggle.com/felipekitamura/head-ct-hemorrhage.

This dataset contains 100 images of normal sections and 100 images of signs of cerebral stroke. However, there is no difference between different types of strokes [12].

Tags are in the CSV file. Each image belongs to a unique person. The main idea behind using such a small dataset is to develop a method that can predict the presence of signs of stroke even with a small amount of baseline data.

Figure 2 shows 20 CT images of the brain in the dataset.

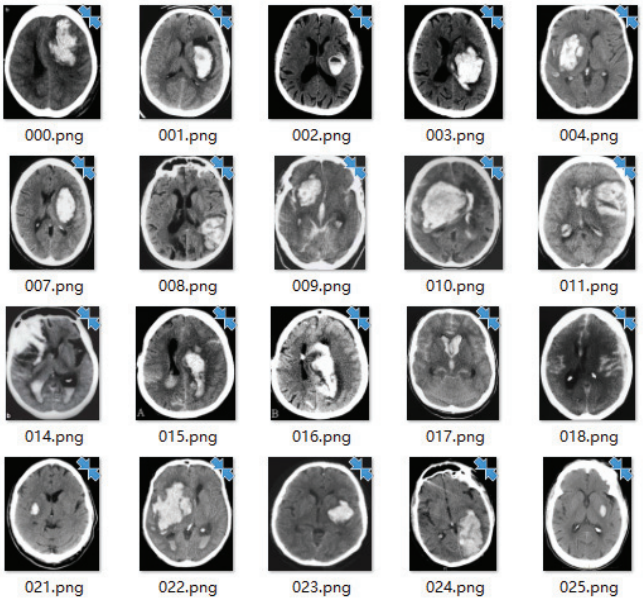

Figure 2. Images of the brain computed tomography results

III. DATA PREPARATION

Before performing any operations with the data, we need to pre-process images in order to improve the estimation of forecast accuracy [13].

In order to obtain higher image fidelity in subsequent training, we examine images length and width to obtain statistics over the set.

Figure 3 shows images distribution in a set by length and width.

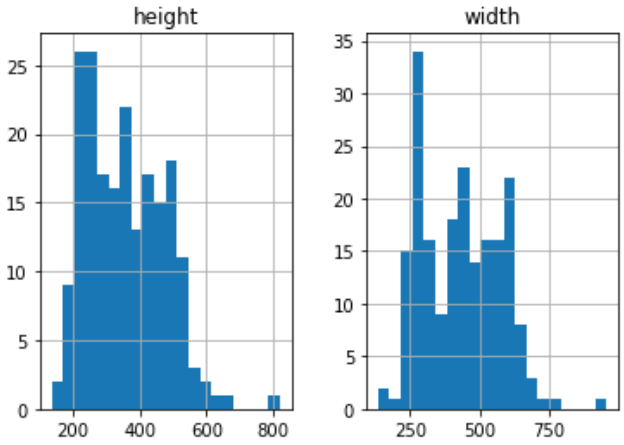

Figure 3. Images distribution by length and width

The description () function is used to display statistics. Results include object counts, means, standard deviations, minimums, maximums, bottom 50 percentile, bottom 25 percentile. 75.50 is both the percentile and median.

Figure 4 shows the results of statistical analysis for length and width.

	height	width
count	200.000000	200.000000
mean	355.505000	433.720000
std	116.785247	142.059481
min	134.000000	135.000000
25%	252.000000	298.750000
50%	345.000000	435.500000
75%	447.250000	554.000000
max	821.000000	957.000000

Figure 4. Results of set analysis in terms of length and width

Before training the model, you should make all images the same size 128 * 128. A simple principle is applied here: the smaller the image size, the faster the training will take place, more objects will be processed, the less likely it is to retrain, but all this is associated with an obvious loss of information.

```
images = np.array([cv2.resize(image, (128, 128)) for image in images])
```

If the error is large, then you should use larger images and either stretch the small images (and lose significantly in quality), or discard them completely and expose network to the risk of overfitting.

Figure 5 shows images with a size of 128 * 128 after processing.

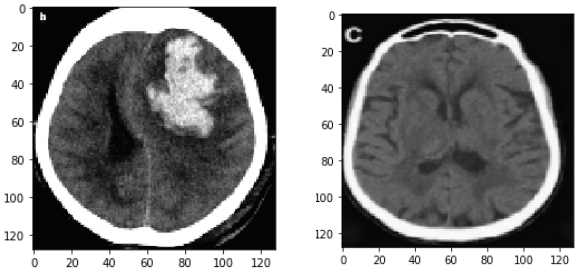

Figure 5. Images 128 * 128

IV. WORKING WITH INVERTED IMAGES

We can also improve the model by adding the ability to work with inverted images. It does not matter how computed tomography is viewed, cerebral hemorrhage can and should be diagnosed in any case. Combined with small capacity of dataset, by adding inverted images to it, we can significantly improve the model accuracy.

Figure 6 shows the results of set images rotating by 90 degrees.

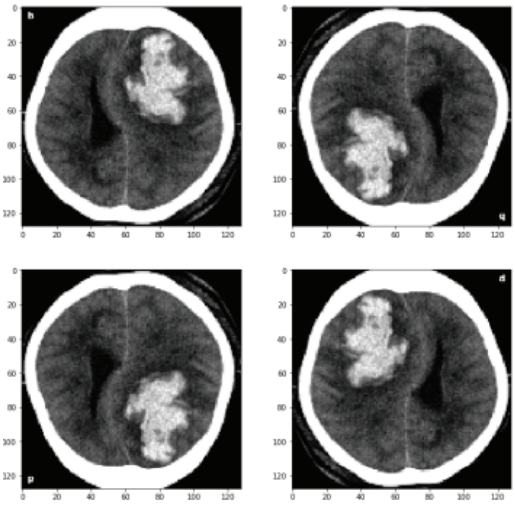

Figure 6. Images are rotated 90°

Using three functions from the library, figure, enumerate and subplot, all preprocessed images are processed and the results are added to the set.

```
plt.figure(figsize=(12, 12))
for i, flip in enumerate([None, -1, 0, 1]):
plt.subplot(221 + i)
if flip is None:
plt.imshow(images[0])
else:
plt.imshow(cv2.flip(images[0], flip)
```

V. IMPROVING IMAGES

Image enhancement [14,15] is implemented using the ImageDataGenerator function, which supports real-time data enhancement. During training, the function will generate data indefinitely until it reaches the specified number of epochs. We improved the image by increasing horizontal vertical deflection by 90° and 180°, horizontal and vertical offset by 0.05 logical values when lifting data, increasing rejection ratio and fixing the image channel size position (128, 128, 3).

```
train_image_data = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.,
    zoom_range=0.05,
    rotation_range=180,
    width_shift_range=0.05,
    height_shift_range=0.05,
    horizontal_flip=True,
    vertical_flip=True,
    fill_mode='constant',
    cval=0)
validation_image_data = ImageDataGenerator(
  rescale=1./255,
  shear_range=0.,
  zoom_range=0.05,
  rotation_range=90,
  width_shift_range=0.05,
  height_shift_range=0.05,
  horizontal_flip=True,
  vertical_flip=True,
  fill_mode='constant',
      cval=0)
```

Figure 7 shows the results of enhancing the images of a set using built-in functions.

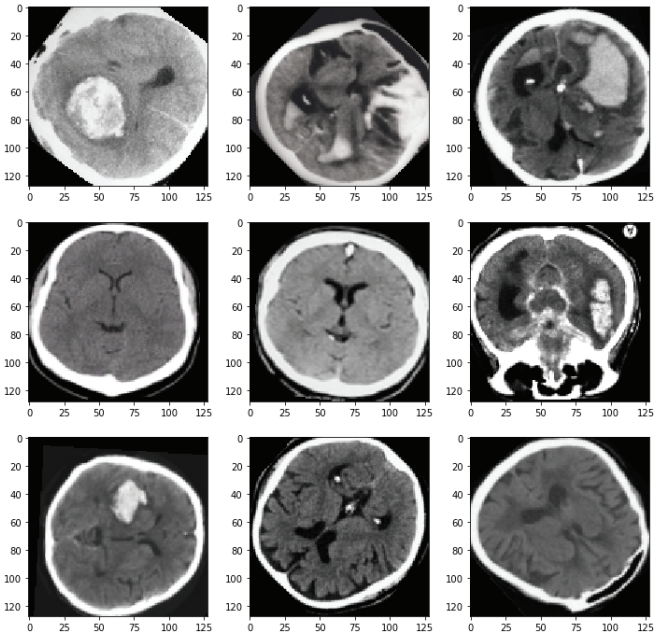

Figure 7. Improved set images

VI. MODEL BUILDING

The SNN model uses a small fixed-size convolution kernel (3 * 3). Convolution pool structure of three convolution layers with two pooling layers. The Sigmoid activation function is used on the output of the convolution layer. Simplified calculation and added Drop layer to prevent model overlap.

Using the summary () function, you can see the complete structure of the network convolutional layer model. The results are shown in Figure 8.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 64, 64, 32)	896
max_pooling2d_1 (MaxPooling2	(None, 32, 32, 32)	0
conv2d_2 (Conv2D)	(None, 16, 16, 32)	9248
max_pooling2d_2 (MaxPooling2	(None, 8, 8, 32)	0
conv2d_3 (Conv2D)	(None, 4, 4, 64)	18496
global_average_pooling2d_1 (	(None, 64)	0
dropout_1 (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 32)	2080
dropout_2 (Dropout)	(None, 32)	0
dense_2 (Dense)	(None, 1)	33

Total params: 30,753
Trainable params: 30,753
Non-trainable params: 0

Figure 8. Structure of network convolutional layer

The first segment of a convolutional network consists of a convolutional layer and a maximum pooling layer. The size of the convolution kernels of these two convolutional layers is 3 * 3, and the number of convolution kernels is 32. The size of the image at the input of the first layer is 128 * 128 * 3, and the size of the output is 64 * 64 * 32. After passing the maximum pooling layer 2 * 2, the output size becomes 32 * 32 * 32, because the step size is 2. Then the result is fed to the second, similar segment of the convolutional network. The sizes of the convolutional kernel of the two convolutional layers are also 3 * 3, but the number of links after the convolutional layer changes, and the final output size becomes 4 * 4 * 64. After processing by the GAP layer to change the data from multidimensional to one-dimensional, and Dropout regularization to prevent overfitting, the generalization of the model is improved, and then the Sigmoid activation function is applied.

The experiment was carried out on a personal computer with Windows 10 operating system; Python programming language was used as an environment for neural network training. Keras library was used for feature extraction, model creation and training.

The set size is 128. After 16 steps, trained model was tested on a set of computed tomograms of brain. The highest accuracy that was achieved is 91.89% at step 16. The results of model testing are shown in Figure 9.

```
def simple_conv_model(input_shape):
 model = Sequential()
    model.add(Conv2D(32, kernel_size=3, strides=2, padding='same', activation='relu', input_shape=input_shape))
    model.add(MaxPooling2D(pool_size=2))
  model.add(Conv2D(32, kernel_size=3, strides=2, padding='same', activation='relu'))
    model.add(MaxPooling2D(pool_size=2))
    model.add(Conv2D(64, kernel_size=3, strides=2, padding='same', activation='relu'))
    model.add(GlobalAveragePooling2D())
    model.add(Dropout(0.4))
    model.add(Dense(32, activation='relu'))
 model.add(Dropout(0.4))
    model.add(Dense(1, activation='sigmoid'))
return model
```

The accuracy and likelihood of retraining the model in the process of working with CT images of the brain change depending on the number of training steps and are presented using Tensorboard visualization in the form of curves of changes in accuracy and loss factor in Figure 10.

```
128/128 [==============================] - 47s 364ms/step - loss: 0.4569 - accuracy: 0.7807 - val_loss: 0.0726 - val_accuracy: 0.7989
Epoch 6/24
128/128 [==============================] - 47s 367ms/step - loss: 0.4236 - accuracy: 0.8070 - val_loss: 0.2790 - val_accuracy: 0.8278
Epoch 7/24
128/128 [==============================] - 47s 365ms/step - loss: 0.3845 - accuracy: 0.8309 - val_loss: 0.0702 - val_accuracy: 0.6722
Epoch 8/24
128/128 [==============================] - 47s 364ms/step - loss: 0.3714 - accuracy: 0.8475 - val_loss: 0.4140 - val_accuracy: 0.8022
Epoch 9/24
128/128 [==============================] - 47s 365ms/step - loss: 0.3250 - accuracy: 0.8674 - val_loss: 0.1843 - val_accuracy: 0.8633
Epoch 10/24
128/128 [==============================] - 47s 366ms/step - loss: 0.2873 - accuracy: 0.8826 - val_loss: 0.6402 - val_accuracy: 0.8444
Epoch 11/24
128/128 [==============================] - 47s 367ms/step - loss: 0.2711 - accuracy: 0.8931 - val_loss: 0.1948 - val_accuracy: 0.8811
Epoch 12/24
128/128 [==============================] - 47s 367ms/step - loss: 0.2410 - accuracy: 0.9079 - val_loss: 0.0430 - val_accuracy: 0.8689
Epoch 13/24
128/128 [==============================] - 47s 368ms/step - loss: 0.2182 - accuracy: 0.9160 - val_loss: 1.2892e-04 - val_accuracy: 0.9011
Epoch 14/24
128/128 [==============================] - 47s 366ms/step - loss: 0.2026 - accuracy: 0.9236 - val_loss: 0.0218 - val_accuracy: 0.8833
Epoch 15/24
128/128 [==============================] - 47s 364ms/step - loss: 0.1930 - accuracy: 0.9289 - val_loss: 0.0669 - val_accuracy: 0.8622
Epoch 16/24
128/128 [==============================] - 47s 365ms/step - loss: 0.1727 - accuracy: 0.9360 - val_loss: 0.0104 - val_accuracy: 0.8422
Epoch 17/24
128/128 [==============================] - 47s 365ms/step - loss: 0.1778 - accuracy: 0.9360 - val_loss: 0.1257 - val_accuracy: 0.9267
Epoch 18/24
128/128 [==============================] - 47s 365ms/step - loss: 0.1543 - accuracy: 0.9436 - val_loss: 0.0052 - val_accuracy: 0.8856
Epoch 19/24
128/128 [==============================] - 47s 364ms/step - loss: 0.1531 - accuracy: 0.9438 - val_loss: 0.2364 - val_accuracy: 0.9200
Epoch 20/24
128/128 [==============================] - 47s 365ms/step - loss: 0.1320 - accuracy: 0.9537 - val_loss: 0.0091 - val_accuracy: 0.8956
Epoch 21/24
128/128 [==============================] - 47s 364ms/step - loss: 0.1265 - accuracy: 0.9525 - val_loss: 0.5809 - val_accuracy: 0.8689
Epoch 22/24
128/128 [==============================] - 47s 364ms/step - loss: 0.1197 - accuracy: 0.9551 - val_loss: 0.0222 - val_accuracy: 0.8978
Epoch 23/24
128/128 [==============================] - 47s 364ms/step - loss: 0.1174 - accuracy: 0.9586 - val_loss: 2.0594e-05 - val_accuracy: 0.9222
Epoch 24/24
128/128 [==============================] - 47s 364ms/step - loss: 0.1100 - accuracy: 0.9618 - val_loss: 0.2535 - val_accuracy: 0.8989
```

Figure 9. Model test results

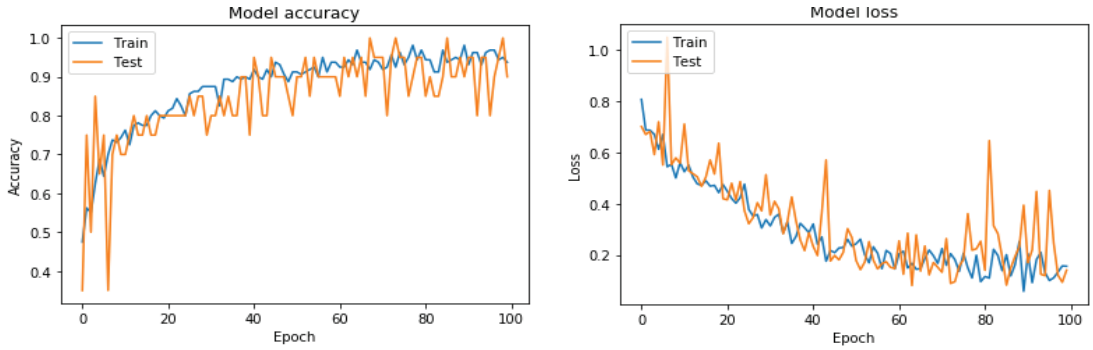

Figure 10. Graphs of changes in the values of accuracy and loss factor

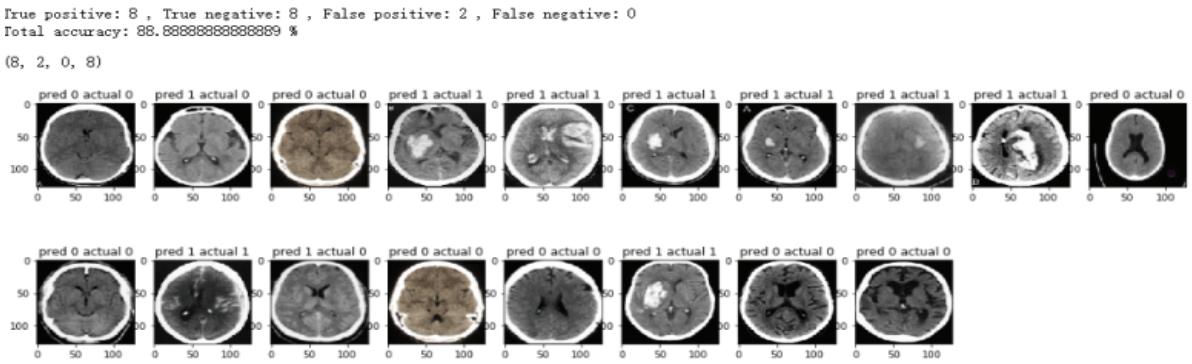

Figure 11. Model results

VII. LTS

The model showed 89% accuracy on the test set. The results obtained indicate full applicability of created model for predicting presence of stroke signs in the brain from computed tomography images. Figure 11 shows the results of created model applying to the test set of images.

CONCLUSION

A predictive model for the presence of signs of cerebral stroke on computed tomography images using a convolutional neural network is proposed and designed. Dataset contained images of normal brains and brains of stroke patients. Prediction accuracy using the model reaches 90%. It is shown that software image enhancement leads to certain effects.

REFERENCES

1. A.D. Bykov, V.I. Voronov. Classification of hydraulic system states by machine learning methods. Information Society Technologies. *XIII international industrial scientific and technical conference*. 2019. P. 403-406.

2. L.I. Voronova, V.I. Voronov. Machine Learning: Regression Data Mining Techniques: Study Guide. MTUCI, 2019. 81 p.

3. A.A. Yezhov, L.I. Voronova, V.I. A.A. Voronov, Goncharenko, M.D. Artemov. A program to support non-verbal communication using machine learning based on a convolutional neural network. Certificate of registration of the computer program ru 2019610179, 09.01.2019. Application No. 2018664377 dated 12/13/2018.

4. A.A. Goncharenko, L.I. Voronova, V.I. Voronov, A.A. Yezhov, M.D. Artemov. A software package for managing data in an information and communication system of social accessibility for people with hearing disabilities. Certificate of registration of the computer program ru 2019610962, 01/18/2019. Application No. 2018665275 dated 26.12.2018.

5. M.D. Artemov, L.I. Voronova, V.I. Voronov, A.A. Goncharenko, A.A. Yezhov. A software package for sign language recognition based on structural and parametric adaptation of a convolutional neural network. Certificate of registration of the program for the computer ru 2018666854, 21.12.2018. Application No. 2018664380 dated 13.12.2018.

6. Wey Kunchao. Research on Medical Image Classificationn Method Based on Convolutional Neural Network. Harbin Institute of Technology. February 2018. -TP391.41. -TP183.

7. Fan Wang, Han Zhongan, Gou Fan et al. Convolutional neural network for Chinese character recognition confirmation code [J]. *Computer Engineering and Applications*. 2018. (3). 160-165.

8. Yang Mengzho, Guo Mengjie, Fang Liang. Study of an Image Classification Algorithm Based on the Keras Convolutional Neural Network, Anhui University. February 2018. TP391.41. TP183

9. Adit Deshpande. What is a convolutional neural network. URL: https://habr.com/en/post/309508/ (date accessed: 09/08/2016).

10. Qiy Xin. research [D] image convolution classification algorithm based on neural network. West Northern Pedagogical University. 2018. Krizhevsky A., Sutskever I., Hinton G.E. Imagenet classification with deep convolutional neura1 networks; Advances in neural information processing systems [C]. 2012.

11. Ivan Golikov, Convolutional neural network, part 1: structure, topology, activation functions, and training set. URL: https://habr.com/en/post/348000/ (date accessed: 31.01.2018).

12. N.K. Verma, P. Gupta, P. Agrawal, etc. Medical Image Segmentation Using Improved Mountain Clustering Approach[C] *VI International Conference on Information Technology: New Generations. IEEE Computer Society*. 2009. P. 1307-1312.

13. A. Jyoti, M.N. Mohanty, S.K. Kar, etc. Optimized Clustering Method for CT Брайн Image Segmentation [M] *Proceedings of the 3rd International Conference он Frontiers of Intelligent Computing: Theory and Applications (FICTA)* 2014. Springer International Publishing. 2014. P. 317-324.

14. .A.V. Doronicheva, S.Z. Savin. Methods of recognition of medical images for the tasks of computer automated diagnostics. *Modern problems of science and education*. 2014. No. 4.