ISSN 2079-3316 **PROGRAM SYSTEMS: THEORY AND APPLICATIONS** **vol. 13**, No 3(54), pp. 45–59
Research Article ARTIFICIAL INTELLIGENCE, INTELLIGENT SYSTEMS, NEURAL NETWORKS

UDC 004.932.75'1+004.89
10.25209/2079-3316-2022-13-3-45-59

Using a convolutional neural network to recognize text elements in poor quality scanned images

Igor Victorovich **Vinokurov**
Financial University under the Government of the Russian Federation, Moscow, Russia
✉*igvvinokurov@fa.ru* *(learn more about the author on p. 59)*

Abstract. The paper proposes a method for recognizing the content of scanned images of poor quality using convolutional neural networks (CNNs). The method involves the implementation of three main stages.

At the first stage, image preprocessing is implemented, which consists of identifying the contours of its alphabetic and numeric elements and basic punctuation marks.

At the second stage, the content of the image fragments inside the identified contours is sequentially fed to the input of the CNN, which implements a multiclass classification.

At the third and final stage, the post-processing of the set of SNA responses and the formation of a text document with recognition results are implemented.

An experimental study of all stages was carried out in Python using the Keras deep learning libraries and OpenCV computer vision and showed fairly good results for the main types of deterioration in the quality of a scanned image: geometric distortions, blurring of borders, the appearance of extra lines and spots during scanning, etc.

Key words and phrases: image processing, convolutional neural network, Python, Keras, OpenCV

2020 *Mathematics Subject Classification*: 68T20; 68T07, 68T45

For citation: Igor V. Vinokurov. *Using a convolutional neural network to recognize text elements in poor quality scanned images* // Program Systems: Theory and Applications, 2022, **13**:3(54), pp. 45–59. `http://psta.psiras.ru/read/psta2022_3_45-59.pdf`

Эта статья по-русски: *http://psta.psiras.ru/read/psta2022_3_29-43.pdf*

Introduction

The task of text recognition on scanned images is quite relevant due to the need to automate the processing of various types of reports, questionnaires, surveys, etc. However, in some cases, the quality of the scanned image may be low due to the darkening of some areas of the image, the presence of arbitrary handwritten marks on the scanned or original images, etc. One possible approach to solving this problem is to use data science methods, since the problem of image element recognition is a typical multiclass classification problem that can be effectively solved using CNN [1]. An example is handwritten digit recognition using a CNN trained on the MNIST [2] dataset. A certain recognition efficiency is achieved due to a large set of options for writing numbers and the presence of CNN models that have proven their effectiveness for solving this and similar tasks. However, as experimental studies have shown, CNNs trained on MNIST do not allow us to recognize numbers on typewritten documents with acceptable accuracy. The same applies to [3, 4] alphabet letters.

An increase in recognition accuracy can be achieved either by using different CNN models, as done in [5, 6], by applying methods for processing the original image: localization, blurring, segmentation and restoration [7, 8], additions source data [9] and use of various information processing methods implemented in the libraries of modern programming languages. [10] Given the lack of formalized methods for synthesizing CNNs in data science, it seems inappropriate to limit ourselves to the selection of their models for solving practical problems. The use of exclusively image processing methods, implemented, among other things, in libraries of programming languages, is also ineffective. Of particular practical interest may be the combination of these two approaches proposed in this paper: identifying the characteristic features of text elements using CNN, processing the original image and recognition results.

The method proposed in this paper is based on the use of a CNN model that implements the classification of selected fragments of a scanned image. It should be noted that, in addition to CNN, other data science methods can be used to solve the multiclass classification problem: decision tree (DT), k-nearest neighbors (KNN), linear discriminant analysis (LDA),

Gaussian naive Bayes method (GNB) and support vector machine (SVM) [11]. However, as experimental studies have shown, it is the use of CNN that gives greater efficiency in solving the classification problem due to the selection of various models and data sets for their training. The study of all CNN models in this work was carried out using a specialized software system (PS) developed in Python using the Keras [12] deep learning libraries, OpenCV [13] computer vision, and PIL [14] image processing libraries.

1. Image Preprocessing

The main goal of this stage is to find the contours of text elements on the scanned image: letters, numbers and basic punctuation marks. The darkening of some areas and the presence of arbitrary handwritten marks and lines that occur due to significant physical deformation of the storage medium can lead to distortion of text elements on the scanned image. In addition, documents may contain many unrecognizable elements, such as vertical and horizontal table lines, decorations, and the like. As a result, the stage of image preprocessing should consist in removing all unnecessary elements from the scanned image and obtaining the cleanest and most contrasting image, which allows detecting the contours of all text elements for their subsequent classification using CNN.

One way to increase the contrast of a scanned image is to convert it to grayscale and then apply a Gaussian blur and a threshold function to the resulting image. These actions in Python are implemented by sequentially calling the functions *cvtColor()*, *GaussianBlur()* and *threshold()* of the OpenCV[15] library and allow you to lighten dark areas to a certain extent and remove all low-contrast extra elements from the scanned image.

Extra lines (mostly horizontal and vertical) can be removed by searching for structural elements of the image and then applying morphological operations with the detection kernels of the corresponding line types: functions *getStructuringElement()* and *morphologyEx()* OpenCV libraries respectively [15]. The result of applying the above OpenCV functions to brighten the scanned image and remove horizontal and vertical

(*a*) extra lines removing

(*b*) contours of text elements

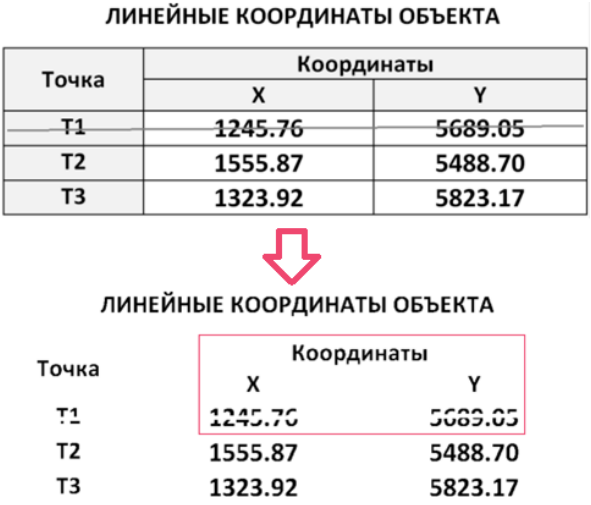
FIGURE 1. Good quality image processing

lines in a tabular presentation of information is shown in Figure 1*a*. The settings for these functions are determined by the quality of the original document and the scanning process.

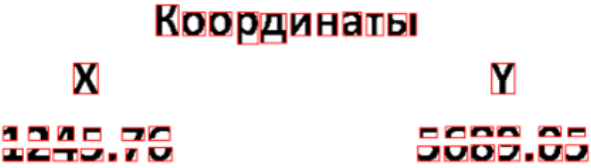
The image obtained in this way makes it possible to obtain the contours of all text elements for their classification using CNN, Figure 1*b*.

The *findContours()* function of the OpenCV [**13**] library was used to find the contours of text elements.

However, the removal of a straight or curly line passing through the text element of the image leads to distortion or segmentation of the latter,



(*a*) line passing through text removing



(*b*) contours of segmented text elements

FIGURE 2. Poor quality image processing

which makes it difficult to further classify it, as we see on Figure 2.

It is the presence of segmented text elements that determined the choice of CNNs capable of classifying fragmented images with sufficiently high accuracy based on stable features of the latter. The paper considered elements of the text with a fragmentation of no more than 10–15%. Immediately prior to CNN classification, all segmented text element outlines are converted into a single 20×25 pixel outline using PIL [**13**] functions. In addition, in some cases, it is possible to restore fragmented images as a result of their skeletonization by calling the functions *erode()* and *dilate()* of the OpenCV [**15**] library, which improves classification accuracy, see Figure 3.

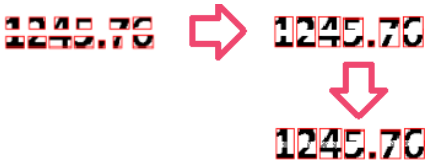


FIGURE 3. Contour content of text elements for their classification on CNN

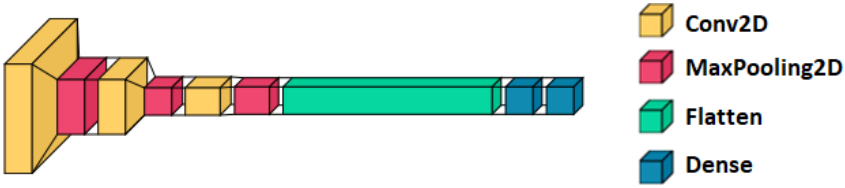


FIGURE 4. CNN organization

2. CNN Model

The formation of almost any CNN model is heuristic in nature and consists in choosing its parameters, structure, and data sets for training and testing.

As noted above, the Keras [12, 16] deep learning library was used to create a CNN model. As a result of experimental studies, the following CNN structure was chosen: 3 convolutional layers (*Conv2D*), 3 pooling layers (*MaxPooling2D*), 1 linearizing layer (*Flatten*) and 2 fully connected layers (*Dense*), see Figure 4.

The description of the CNN layers is shown in Figure 5.

The activation functions of all CNN neurons are *«sigmoid»*. Compilation and training parameters are chosen as standard for multiclass classification: optimizer *«adam»*, loss function *«categorical-crossentropy»*, metric *«accuracy»*.

According to the requirements of Keras, the preparation of datasets for the implementation of a multiclass classification CNN consists in the formation of images of text elements and their correlation with the corresponding image classes [16]. The number of image classes in this work was chosen to be 42, namely: 10 classes for numbers from 0 to 9, 30 classes for the letters of the Russian alphabet, with the exception of the compound letters «ё», «й», «ы» and 2 classes for the main punctuation marks «.» and «,». For each class of digital images, 10 and 5 training and testing images

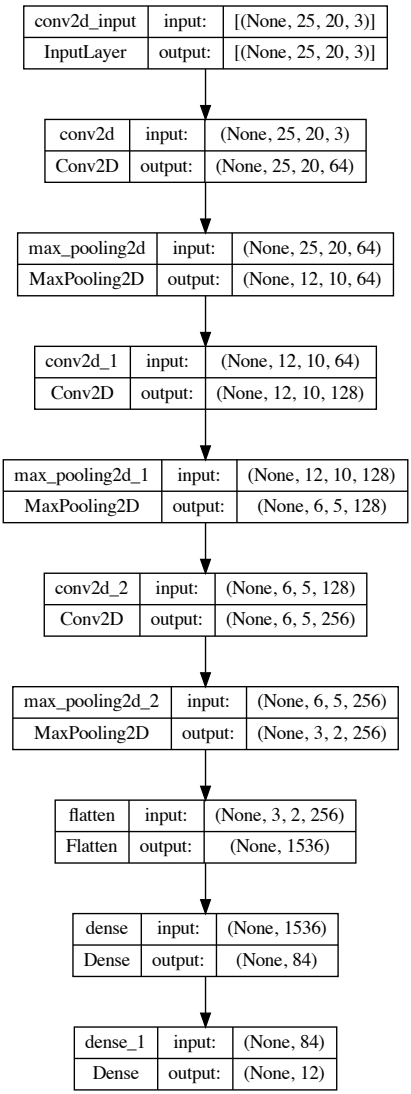
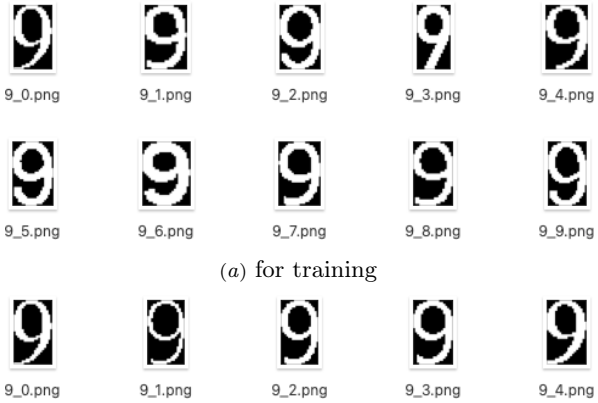


FIGURE 5. CNN layers



(*a*) for training

(*b*) for testing

FIGURE 6. Sample images for learning

were formed, for letter images, 20 and 10, respectively. Lowercase and uppercase letters belong to the same class.

A small number of images is explained by the uniformity of the elements of typewritten documents when using basic fonts. In addition, the presence in the Keras library of the image batch generator *ImageDataGenerator* allows you to get different variations of images through *the use of transformations compression, stretching, tilting, etc.*.

For the formation of training images, fonts were chosen that implement all the main features of displaying text elements: Times.ttf, Calibri.ttf, Arial.ttf, Bahnschrift.ttf, Cambria.ttc, RockwellNova.ttf, Framd.ttf, Tahoma.ttf, Corbel. ttf and Micross.ttf. The test images are generated by the fonts Times.ttf, Segoeuil.ttf, Georgia.ttf, Lucon.ttf and Cour.ttf. For example, the contents of the training and test sets of images for the number «9» are shown in Figure 6.

Before the CNN training process, all images from the training and test sets are resized by the *resize()* function of the OpenCV library to a size of 20×25.

Dependences of the accuracy of the model and its loss function for all training epochs are shown in Figure 7.

Figure 8 shows the resulting values of the loss and accuracy functions for the test data set with 50 training epochs and a data batch size (batch) of 10 were 0.1334 and 0.9653 respectively.

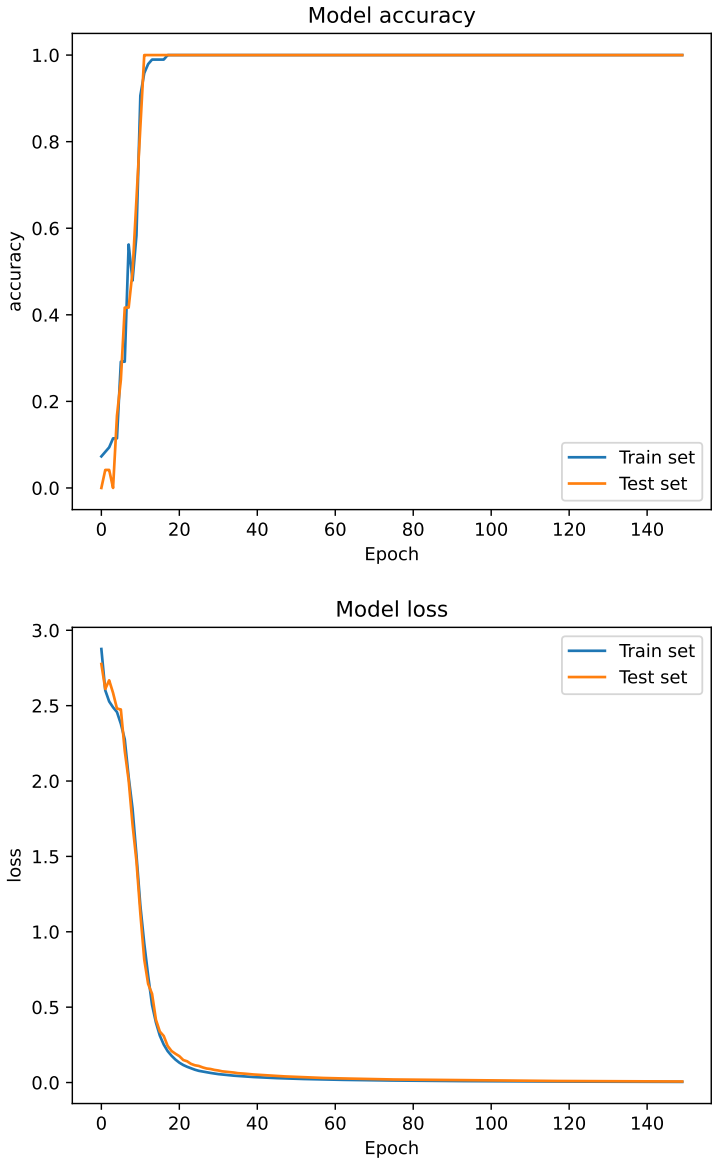


FIGURE 7. CNN model accuracy and loss

```
Epoch 48/50
58/58 [==============================] - 1s 10ms/step - loss: 0.0091 - acc: 1.0000
Epoch 49/50
58/58 [==============================] - 1s 10ms/step - loss: 0.0087 - acc: 1.0000
Epoch 50/50
58/58 [==============================] - 1s 10ms/step - loss: 0.0082 - acc: 1.0000
5/5 [==============================] - 0s 7ms/step - loss: 0.1334 - acc: 0.9653
[0.1333889365196228, 0.9652777910232544]
```

FIGURE 8. Values of loss and accuracy functions for test images

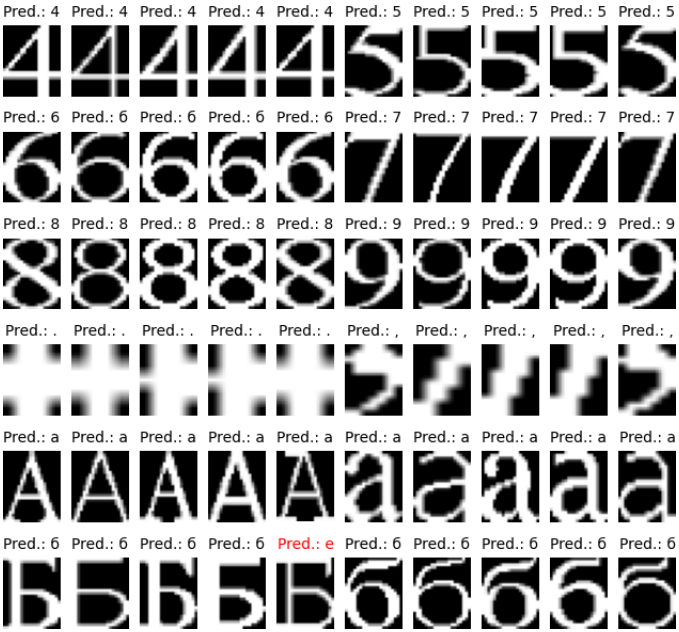


FIGURE 9. CNN predictions of some test images

The classification results for some images from a fairly small test set are shown in Figure 9. For an incorrectly classified image, its title is displayed in red.

3. Image Fragment Recognition with CNN

The purpose of this final step is to recognize CNN text elements whose contours were revealed during the pre-processing of the scanned image. Its implementation consists in recognizing the contents of all the contours

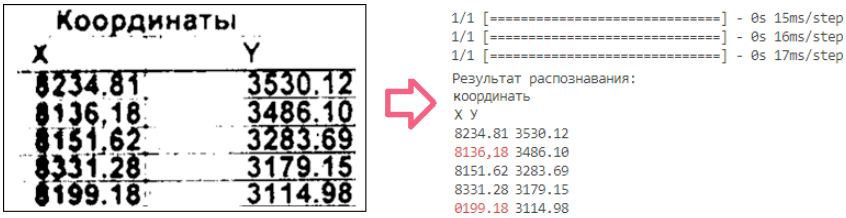


FIGURE 10. Text elements recognition results

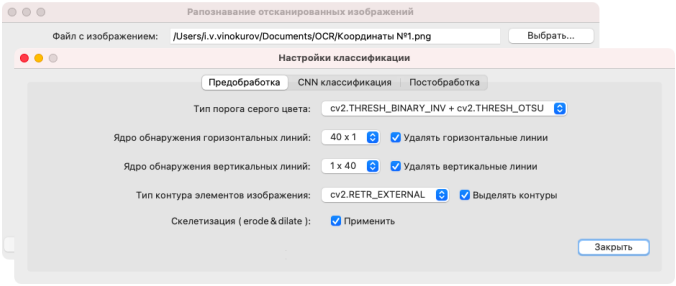


FIGURE 11. Setting the parameters of the preprocessing step

of the text elements identified at the previous stage, in other words, in their correlation with one of the 42 image classes.

As experimental studies have shown, the use of a CNN of any structure does not allow one to recognize the scanned text with 100% accuracy. As a result, at the final stage, depending on the content, the numbers are replaced with letters and vice versa and the resulting text is compared with the results of its recognition using Tesseract OCR from Google [**17**]. The results of recognition of poor quality image content with their subsequent correction are shown in Figure 10. Recognition errors are highlighted in red.

Based on the results of the research, a specialized PS was developed. Python was chosen as the programming language because it implements many libraries for data processing [**18**]. PS allows you to set the parameters of image preprocessing (Figure 11), CNN model compilation options, (Figure 12), post-processing options (Figure 13), and visualize all the main stages of work (Figure 14).

The developed PS is currently used in the branch of the Federal State Budgetary Institution «FKP Rosreestra» in the Kaluga Region to automate

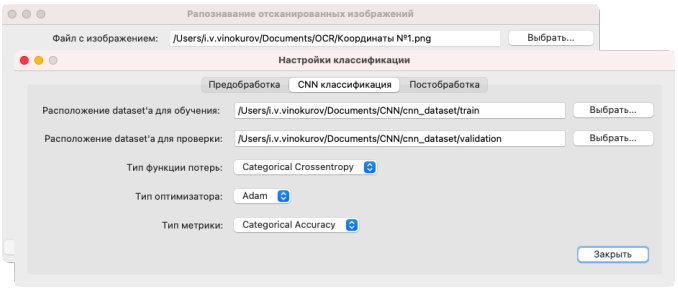


FIGURE 12. Configuring CNN model compilation options

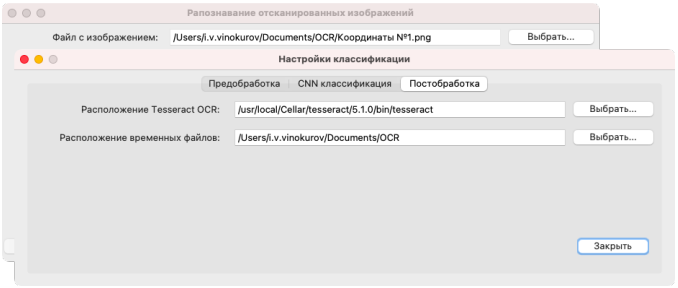


FIGURE 13. Setting up the post-processing step

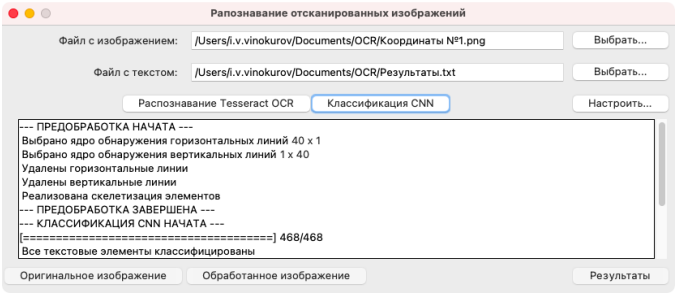


FIGURE 14. Visualization of classification stages

the processing of scanned documents with the coordinates of land plots and capital construction projects.

Conclusion

Based on the mathematical and experimental studies carried out, a software system was developed that implements the recognition of text elements on scanned images of poor quality. PS implements the work of the CNN model, correlating the selected fragments of the original image with the classes of the main elements of the text: letters, numbers and punctuation marks. The selected CNN model, its compilation parameters, pre-processing of the scanned image and processing of the classification results made it possible to obtain a sufficiently high recognition accuracy of about 96% for text elements with fragmentation within 10-15%.

References

[1] A. Chauhan. *Convolutional Neural Networks for multiclass image classification — A beginners guide to understand CNN*, Published in The Startup, 2020. ↑46

[2] J. Brownlee. *How to develop a CNN for MNIST handwritten digit classification*, Machine Learning Mastery, 2019. ↑46

[3] V. Mokin. *MNIST models testing: typographic digits*, Kaggle, 2021. ↑46

[4] K. Y. Chan. *Font recognition using CNN approach*, Final Year Project (Bachelor), Tunku Abdul Rahman University College, 2021. ↑46

[5] Y. Gao, Y. Chen, J. Wang, M. Tang, H. Lu. "Reading scene text with fully convolutional sequence modeling", *Neurocomputing*, **339** (2019), pp. 161–170. ↑46

[6] A. A. Chandio, Mehw. Leghari, Mehj. Leghari, A. H. Jalbani. "Multi-font and multi-size printed Sindhi character recognition using Convolutional Neural Networks", *Pak. J. Engg. Appl. Sci.*, **25** (2019), pp. 36–42. ↑46

[7] G. J. Ansari, J. H. Shah, M. A. Khan, M. Sharif, U. Tariq, T. Akram. "A non-blind deconvolution semi pipelined approach to understand text in blurry natural images for edge intelligence", *Information Processing & Management*, **58**:6 (2021), 102675. ↑46

[8] Zh. Zhong, L. Jin, Z. Feng. "Multi-font printed Chinese character recognition using multi-pooling convolutional neural network", 2015 13th International Conference on Document Analysis and Recognition (ICDAR) (23–26 August 2015, Tunis, Tunisia), 2015, pp. 96–100. ↑46

[9] Ch. Tensmeyer, D. Saunders, T. Martinez. *Convolutional Neural Networks for font classification*, ICDAR 2017, 2017, 7 pp. arXiv 1708.03669 [cs.CV]↑46

[10] J. Charles, P. Y. Simard, P. Viola, J. Rinker. "Text recognition of low-resolution document images", *Eighth International Conference on Document Analysis and Recognition* . V. 2, ICDAR'05 (31 August 2005–01 September 2005, Seoul, South Korea), 2005, ISBN 0-7695-2420-6, pp. 695–699. ↑46

[11] A. Myuller, S. Gvido. *Introduction to Machine Learning with Python. Guide for data scientists*, Vil'yams, M., 2017, ISBN 978-5-9908910-8-1 (In Russian), 394 pp.↑47

[12] A. Dzhulli, S. Pal. *Keras library — deep learning tool*, DMK Press, M., 2017, ISBN 978-5-97060-573-8 (In Russian), 294 pp.↑47, 50

[13] S. Datta. *Learning OpenCV 3 Application Development*, Packt Publiching, 2016, ISBN 978-1784391454, 310 pp.↑47, 48, 49

[14] S. Gruppetta. *Image processing with the Python Pillow Library*, RealPython, 2022. ↑47

[15] S. Dey. *Hands-On Image Processing with Python: Expert techniques for advanced image analysis and effective interpretation of image data*, Packt Publiching, 2018, ISBN 978-1789343731, 492 pp.↑47, 49

[16] V. K. Ayyadevara. *Neural Networks with Keras Cookbook: Over 70 recipes leveraging deep learning techniques across image, text, audio, and game bots*, Packt Publiching, 2019, ISBN 978-1789346640, 568 pp.↑50

[17] A. Rosebrock. *Using Tesseract OCR with Python*, PyimageSearch, 2021. ↑55

[18] U. Makkinni. *Python and data analysis*, DMK Press, M., 2019, ISBN 978-5-97060-590-5 (In Russian), 540 pp.↑55

Received 03.09.2022;
approved after reviewing 05.09.2022;
accepted for publication 15.09.2022.

Recommended by *prof. A. M. Elizarov*

Information about the author:

Igor Victorovich Vinokurov

Candidate of Technical Sciences (PhD), Associate Professor at the Financial University under the Government of the Russian Federation. Research interests: information systems, information technologies, data processing technologies.

0000-0001-8697-1032

e-mail: igvvinokurov@fa.ru

The author declare no conflicts of interests.