ISSN 2079-3316 **ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ** **т. 13**, № 3(54), с. 29–43
научная статья ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ, ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ, НЕЙРОННЫЕ СЕТИ

УДК 004.932.75'1+004.89
10.25209/2079-3316-2022-13-3-29-43

Использование свёрточной нейронной сети для распознавания элементов текста на отсканированных изображениях плохого качества

Игорь Викторович **Винокуров**✉
Финансовый Университет при Правительстве Российской Федерации, Москва, Россия
✉*igvvinokurov@fa.ru* *(подробнее об авторе на с. 42)*

Аннотация. В работе предложен метод распознавания содержимого отсканированных изображений плохого качества с использованием свёрточных нейронных сетей (CNN) и его программная реализация. Метод состоит из 3-х основных этапов.

На первом этапе осуществляется предобработка изображения с целью выявлении контуров его буквенных и цифровых элементов и основных знаков пунктуации.

На втором этапе содержимое фрагментов изображения внутри выявленных контуров последовательно подаётся на вход CNN, реализующую многоклассовую классификацию.

На третьем, заключительном этапе, осуществляется постобработка совокупности ответов CNN и формирование текстового документа с результатами распознавания.

Все этапы реализованы на языке Python с использованием библиотек глубокого обучения Keras, компьютерного зрения OpenCV и обработки изображений PIL. Предлагаемый в работе способ показал достаточно хорошие результаты распознавания для основных типов ухудшения качества отсканированного изображения – геометрических искажений, размытия границ, различных пометок на исходном и отсканированном изображении и т.п.

Ключевые слова и фразы: обработка изображений, свёрточная нейронная сеть, Python, Keras, OpenCV

Для цитирования: Винокуров И.В. *Использование свёрточной нейронной сети для распознавания элементов текста на отсканированных изображениях плохого качества* // Программные системы: теория и приложения. 2022. **Т. 13**. № 3(54). С. 29–43. http://psta.psiras.ru/read/psta2022_3_29-43.pdf

This Article in English: *http://psta.psiras.ru/read/psta2022_3_45-59.pdf*

Введение

Задача распознавания текста на отсканированных изображениях является достаточно актуальной в связи с необходимостью автоматизации обработки различного вида отчётностей, анкет, опросов и т.п. Однако в ряде случаев качество отсканированного изображения может быть невысоким за счёт затемнения некоторых участков изображения, наличия на отсканированном или исходном изображениях произвольных рукописных пометок и т.п. Одним из возможных подходов к решению этой задачи является использование методов Data Science, поскольку распознавание элемента изображения является типичной задачей многоклассовой классификации, эффективно решаемой с использованием CNN [1]. В качестве примера можно привести распознавание рукописных цифр с использованием CNN, обученной на наборе MNIST [2]. Определённая эффективность распознавания достигается за счёт большого набора вариантов написания цифр и наличием моделей CNN, доказавших свою эффективность для решения этой и подобных ей задач. Однако, как показали экспериментальные исследования, CNN, обученные на MNIST, не позволяют с приемлемой точностью распознавать цифры на машинописных документах. То же самое относится и к буквам алфавита [3, 4].

Повышение точности распознавания может быть достигнуто либо за счёт использования различных моделей CNN, как это сделано в [5, 6], применения методов обработки оригинального изображения – локализация, размытие, сегментация и восстановление [7, 8], дополнения исходных данных [9] и использования различных методов обработки информации, реализованных в библиотеках современных языков программирования [10]. Учитывая отсутствие в Data Science формализованных методов синтеза CNN, ограничиваться подбором их моделей для решения практических задач представляется нецелесообразным. Малоэффективным является и использование исключительно методов обработки изображений, реализованных, в том числе, и в библиотеках языков программирования. Определенный практических интерес может представлять предлагаемое в данной работе объединение этих двух подходов – выявление характерных признаков элементов текста с использованием CNN, обработка оригинального изображения и результатов его распознавания.

В основе предлагаемого в данной работе метода лежит использование модели CNN, реализующей классификацию выделенных фрагментов отсканированного изображения. Следует отметить, что для

решения задачи многоклассовой классификации, помимо CNN, могут быть использованы и другие методы Data Science – дерево решений (DT), k-ближайших соседей (KNN), линейный дискриминантный анализ (LDA), гауссовый наивный байесовский метод (GNB) и метод опорных векторов (SVM) [11]. Однако, как показали экспериментальные исследования, именно использование CNN даёт большую эффективность решения задачи классификации за счёт подбора различных моделей и наборов данных для её обучения. Исследование всех моделей CNN в этой работе проводилось с помощью разработанной на языке Python специализированной программной системы (ПС), использующей библиотеки глубокого обучения Keras [12], компьютерного зрения OpenCV [13] и обработки изображений PIL [14].

1. Предварительная обработка изображения

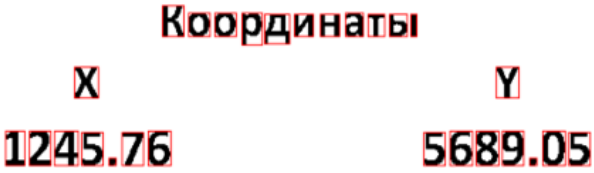
Основной целью этого этапа является нахождение контуров текстовых элементов на отсканированном изображении – букв, цифр и основных знаков пунктуации. Затемнение некоторых участков и наличие произвольных рукописных пометок и линий, возникающих по причине значительной физической деформации носителя информации, может приводить к искажению текстовых элементов на отсканированном изображении. Помимо этого, документы могут содержать множество элементов, не подлежащих распознаванию, например, вертикальные и горизонтальные линии таблиц, элементы декорирования и т.п. Как следствие, этап предварительной обработки изображения должен заключаться в удалении с отсканированного изображения всех лишних элементов и получении максимально чистого и контрастного изображения, позволяющего детектировать контуры всех текстовых элементов для их последующей классификации с помощью CNN.

Одним из способов повышения контрастности отсканированного изображения является преобразование его в градации серого цвета и последующего применения к результирующему изображению гауссова размытия и пороговой функции. Эти действия на языке Python реализуются последовательным вызовом функций cvtColor(), GaussianBlur() и threshold() библиотеки OpenCV[15] и позволяют в определенной мере осветлить затемнённые участки и удалить все слабоконтрастные лишние элементы из отсканированного изображения.

Лишние линии (в основном, горизонтальные и вертикальные) можно удалить поиском структурных элементов изображения и последующим применением морфологических операций с ядрами детектирования

соответствующих типов линий – функции getStructuringElement() и morphologyEx() библиотеки OpenCV соответственно [15]. Результат применения указанных выше функций OpenCV для осветления отсканированного изображения и удаления горизонтальных и вертикальных линий при табличном представлении информации показан на рисунке 1*а*. Параметры этих функций определяются качеством исходного документа и процесса сканирования.

(*а*) удаление лишних линий



(*б*) контуры элементов текста

Рисунок 1. Обработка изображения хорошего качества

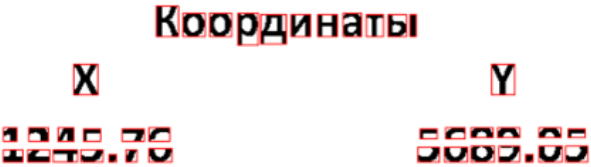
Полученное таким образом изображение позволяет получить контуры всех текстовых элементов для их классификации с помощью CNN, рисунок 1*б*.

Для нахождения контуров текстовых элементов использовалась функция findContours() библиотеки OpenCV [13].

Однако удаление прямой или фигурной линии, проходящей по тек-

стовому элементу изображения, как на рисунке 2 приводит к искажению или сегментации последнего. Это затрудняет его последующую классификацию.

(*а*) удаление линии, проходящей по тексту



(*б*) Контуры сегментированных элементов текста

Рисунок 2. Обработка изображения плохого качества

Именно наличие сегментированных текстовых элементов и определило выбор CNN, способных с достаточно большой точностью классифицировать фрагментированные изображения исходя из устойчивых признаков последних. В работе рассматривались элементы текста с фрагментацией не более 10–15%. Непосредственно перед классификацией CNN, все контуры сегментированного текстового элемента, посредством функций библиотеки PIL [**13**], преобразуются в один размером 20×25 пикселей. Помимо этого, в ряде случаев, как, например на рисунке 3, удаётся восстановить фрагментированные изображения в результате их скелетизации вызовом функций erode() и

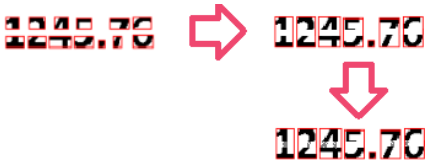


Рисунок 3. Содержимое контуров элементов текста для их классификации на CNN

dilate() библиотеки OpenCV [15], что позволяет повысить точность классификации.

2. Модель CNN

Формирование практически любой модели CNN носит эвристический характер и заключается в выборе параметров и структуры её слоёв и наборов данных для обучения и тестирования.

Как уже было отмечено выше, в данной работе для создания модели CNN использовалась библиотека глубокого обучения Keras [12, 16]. В результате экспериментальных исследований была выбрана следующая структура CNN – 3 свёрточных слоя (Conv2D), 3 слоя пуллинга (MaxPooling2D), 1 линеаризирующий слой (Flatten) и 2 полносвязных слоя (Dense), рисунок 4.

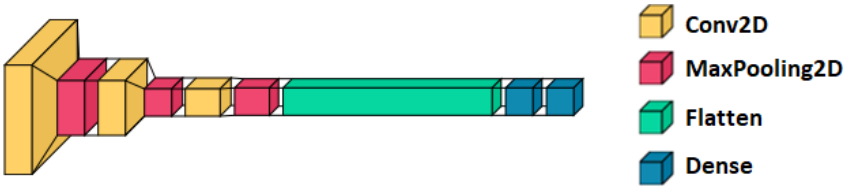


Рисунок 4. Структура CNN

Описание слоёв CNN приведено на рисунке 5.

Функции активации всех нейронов CNN – *«sigmoid»*. Параметры компиляции и обучения выбраны стандартными для многоклассовой классификации: оптимизатор – *«adam»*, функция потерь – *«categorical-crossentropy»*, метрика – *«accuracy»*.

В соответствии с требованиями Keras, подготовка наборов данных для реализации CNN многоклассовой классификации заключается в формировании изображений текстовых элементов и соотнесении их с соответствующими классам изображений [16]. Количество классов

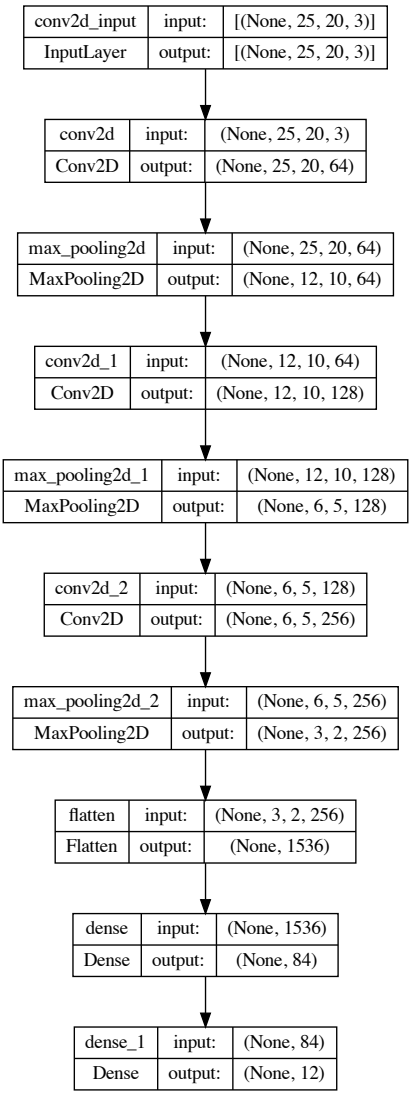
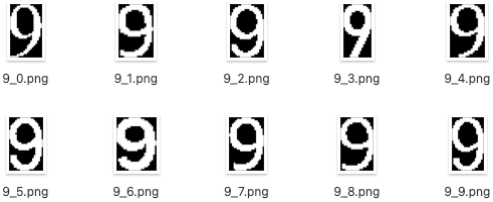


Рисунок 5. Слои CNN

изображений в данной работе было выбрано равным 42 – 10 классов для цифр от 0 до 9, 30 классов для букв русского алфавита, за исключением составных букв «ё», «й», «ы» и 2 класса для основных знаков пунктуации «.» и «,». Для каждого класса цифровых изображений сформировано по 10 и 5 обучающих и тестирующих изображений, для буквенных – по 20 и 10 соответственно. Строчные и прописные буквы относятся к одному классу. Небольшое количество изображений объясняется однотипностью элементов машинописных документов при использовании основных шрифтов. Помимо этого, наличие в библиотеке Keras генератора пакетов изображений *ImageDataGenerator* позволяет получить различные вариации изображений за счёт использования *преобразований сжатия, растяжения, наклона и т.п*.

Для формирования обучающих изображений были выбраны шрифты, реализующие все основные особенности отображения текстовых элементов – Times.ttf, Calibri.ttf, Arial.ttf, Bahnschrift.ttf, Cambria.ttc, RockwellNova.ttf, Framd.ttf, Tahoma.ttf, Corbel.ttf и Micross.ttf. Тестирующие изображения формируются шрифтами Times.ttf, Segoeuil.ttf, Georgia.ttf, Lucon.ttf и Cour.ttf. Для примера, содержимое обучающего и тестового наборов изображений для цифры «9» приведено на рисунке 6.



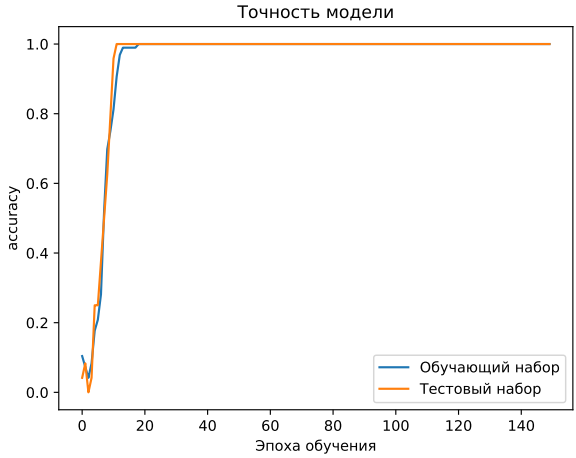
(*а*) для обучения

(*б*) для тестирования

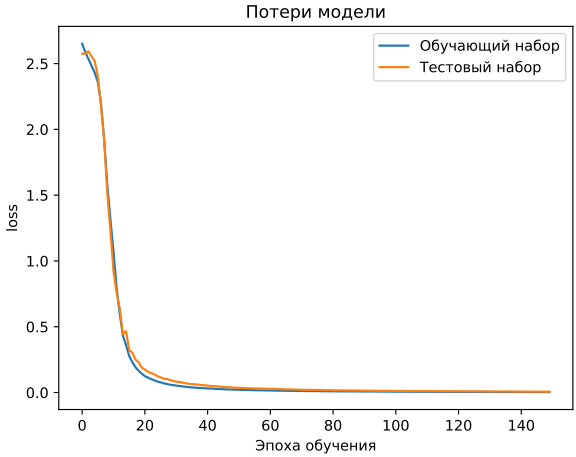
Рисунок 6. Примеры изображений для обучения

Перед процессом обучения CNN все изображения из обучающего и тестового наборов функцией resize() библиотеки OpenCV приводятся к размеру 20×25.

Зависимости точности модели и её функции потерь для всех эпох обучения приведены на рисунке 7.



(*а*) Точность



(*б*) Потери

Рисунок 7. Точность и потери модели CNN

На рисунке 8 показаны результирующие значения функций потерь и точности для тестового набора данных. При 50 эпохах обучения и размере пакета данных (batch) равном 10, составили 0.1334 и 0.9653

```
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]
```

Рисунок 8. Значения функций потерь и точности модели CNN для тестовых изображений

соответственно.

Результаты классификации некоторых изображений из достаточно малого тестового набора приведены на рисунке 9. Для неверно

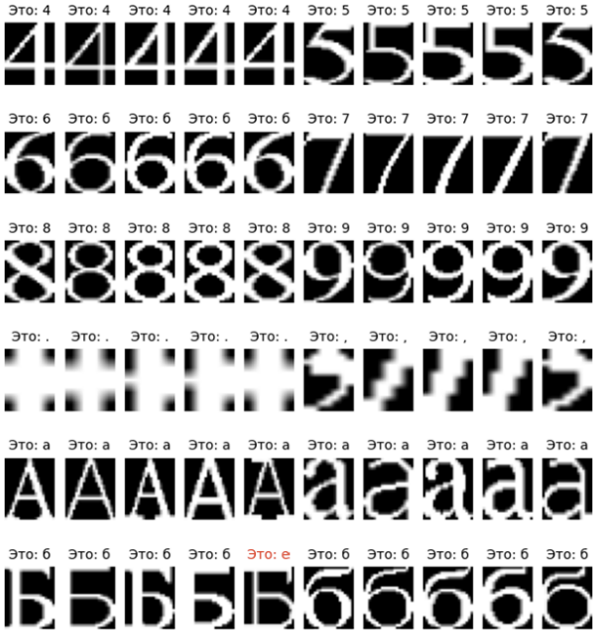


Рисунок 9. Классификация CNN некоторых тестовых изображений

классифицированного изображения его заголовок отображается красным цветом.

3. Распознавание фрагментов изображения с помощью CNN

Цель этого заключительного этапа – распознавание CNN текстовых элементов, контуры которых были выявлены на этапе предварительной обработки отсканированного изображения. Его реализация заключается в распознавании содержимого всех выявленных на предыдущем этапе контуров элементов текста, иными словами – в соотнесении их с одним из 42-х классов изображений.

Как показали экспериментальные исследования, использование CNN любой структуры не позволяет распознать отсканированный текст с 100% точностью. Как следствие, на заключительном этапе, в зависимости от контента, реализуется замена цифр на буквы и наоборот и сопоставление полученного текста с результатами его распознавания посредством Tesseract OCR от Google [17]. Результаты распознавания содержимого изображения плохого качества с их последующей коррекцией приведены на рисунке 10. Красным цветом

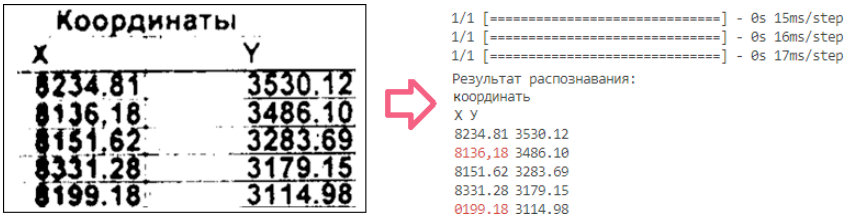
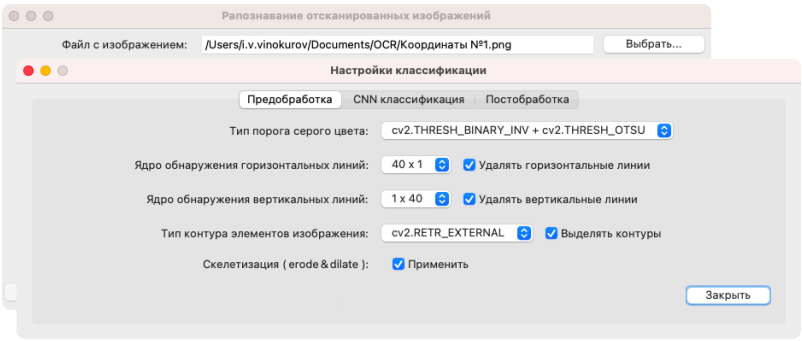


Рисунок 10. Результаты распознавания элементов текста

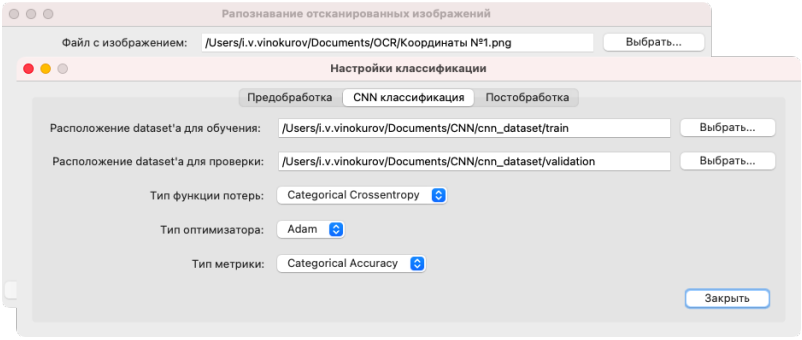
выделены ошибки распознавания.

По результатам проведённых исследований было разработано специализированная ПС. Языком программирования выбран Python, поскольку в нем реализовано множество библиотек для обработки данных [18]. ПС позволяет задавать параметры предварительной обработки изображения (рисунок 11*а*), параметры компиляции модели CNN (рисунок 11*б*), параметры постобработки (рисунок 11) и визуализировать все основные этапы работы (рисунок 12).

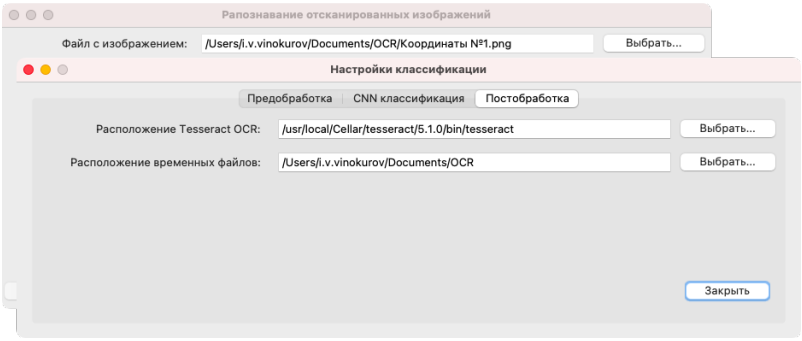
Разработанная ПС в настоящее время используется в филиале ФГБУ «ФКП Росреестра» по Калужской области для автоматизации обработки отсканированных документов с координатами земельных участков и объектов капитального строительства.



(*а*) настройка параметров этапа предобработки



(*б*) настройка параметров компиляции модели CNN



(*в*) настройка этапа постобработки

Рисунок 11. Настройки специализированной ПС

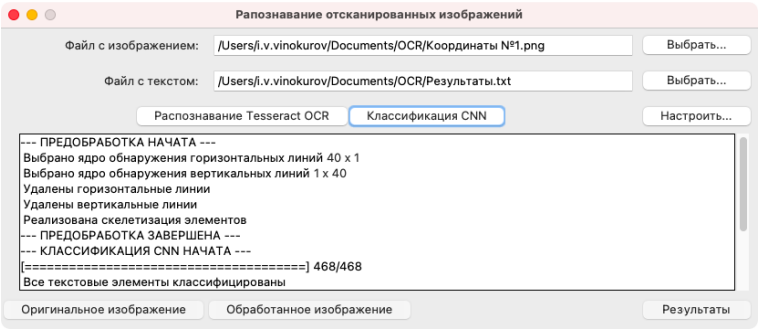


Рисунок 12. Визуализация этапов классификации

Заключение

На основании проведенных математических и экспериментальных исследований разработана ПС, реализующая распознавание элементов текста на отсканированных изображениях плохого качества. ПС реализует работу модели CNN, соотносящий выделенные фрагменты исходного изображения с классами основных элементов текста –букв, цифр и знаков препинания. Выбранная модель CNN, параметры её компиляции, предварительная обработка отсканированного изображения и обработка результатов классификации позволили получить достаточно высокую точность распознавания порядка 96% для элементов текста с фрагментацией в пределах 10–15%.

Список литературы

[1] Chauhan A. *Convolutional Neural Networks for multiclass image classification — A beginners guide to understand CNN*, Published in The Startup.– 2020. ↑30

[2] Brownlee J. *How to develop a CNN for MNIST handwritten digit classification*, Machine Learning Mastery.– 2019. ↑30

[3] Mokin V. *MNIST models testing: typographic digits*, Kaggle.– 2021. ↑30

[4] Chan K. Y. *Font recognition using CNN approach*, Final Year Project (Bachelor).– Tunku Abdul Rahman University College.– 2021. ↑30

[5] Gao Y., Chen Y., Wang J., Tang M., Lu H. *Reading scene text with fully convolutional sequence modeling* // Neurocomputing.– 2019.– Vol. **339**.– pp. 161–170. ↑30

[6] Chandio A. A., Leghari Mehw., Leghari Mehj., Jalbani A. H. *Multi-font and multi-size printed Sindhi character recognition using Convolutional Neural Networks* // Pak. J. Engg. Appl. Sci.– 2019.– Vol. **25**.– pp. 36–42. ↑30

[7] Ansari G. J., Shah J. H., Khan M. A., Sharif M., Tariq U., Akram T. *A non-blind deconvolution semi pipelined approach to understand text in blurry natural images for edge intelligence* // Information Processing & Management.– 2021.– Vol. **58**.– No. 6.– 102675. ↑30

[8] Zhong Zh., Jin L., Feng Z. *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. ↑30

[9] Tensmeyer Ch., Saunders D., Martinez T. *Convolutional Neural Networks for font classification*, ICDAR 2017.– 2017.– 7 pp. arXiv 1708.03669 [cs.CV] ↑30

[10] Charles J., Simard P. Y., Viola P., Rinker J. *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. ↑30

[11] Мюллер А., Гвидо С. *Введение в машинное обучение с помощью Python. Руководство для специалистов по работе с данными.*– М.: Вильямс.– 2017.– ISBN 978-5-9908910-8-1.– 394 с. ↑31

[12] Джулли А., Пал С. *Библиотека Keras – инструмент глубокого обучения.*– М.: ДМК Пресс.– 2017.– ISBN 978-5-97060-573-8.– 294 с. ↑31, 34

[13] Datta S. *Learning OpenCV 3 Application Development.*– Packt Publiching.– 2016.– ISBN 978-1784391454.– 310 pp. ↑31, 32, 33

[14] Gruppetta S. *Image processing with the Python Pillow Library*, RealPython.– 2022. URL ↑31

[15] Dey S. *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. ↑31, 32, 34

[16] Ayyadevara V. K. *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. ↑34

[17] Rosebrock A. *Using Tesseract OCR with Python*, PyimageSearch.– 2021. URL ↑39

[18] Маккинни У. *Python и анализ данных.*– М.: ДМК Пресс.– 2019.– ISBN 978-5-97060-590-5.– 540 с. ↑39

Поступила в редакцию 28.07.2022;
одобрена после рецензирования 31.08.2022;
принята к публикации 15.09.2022.

Рекомендовал к публикации *д.ф.-м.н. А. М. Елизаров*

Информация об авторе:

Игорь Викторович Винокуров

Кандидат технических наук (PhD), ассоциированный профессор в Финансовом Университете при Правительстве Российской Федерации. Область научных интересов: информационные системы, информационные технологии, технологии обработки данных.

0000-0001-8697-1032

e-mail: igvvinokurov@fa.ru

Автор заявляет об отсутствии конфликта интересов.