Received January 10, 2020, accepted February 11, 2020, date of publication February 14, 2020, date of current version February 25, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2974051

Two-Step CNN Framework for Text Line Recognition in Camera-Captured Images

YULIA S. CHERNYSHOVA[1,2], **ALEXANDER V. SHESHKUS**[1,2], **AND VLADIMIR V. ARLAZAROV**[3,4]
[1]Federal Research Center "Computer Science and Control" of RAS, 119333 Moscow, Russia
[2]Smart Engines Service LLC, 117312 Moscow, Russia
[3]Institute for Information Transmission Problems (Kharkevich Institute) RAS, 127051 Moscow, Russia
[4]Moscow Institute of Physics and Technology (National Research University), 141701 Moscow, Russia

Corresponding author: Yulia S. Chernyshova (chernyshovayulia07@gmail.com)

This work was supported in part by the RFBR According to the Research Project 17-29-03170 and Project 17-29-03236.

ABSTRACT In this paper, we introduce an "on the device" text line recognition framework that is designed for mobile or embedded systems. We consider per-character segmentation as a language-independent problem and individual character recognition as a language-dependent one. Thus, the proposed solution is based on two separate artificial neural networks (ANN) and dynamic programming instead of employing image processing methods for the segmentation step or end-to-end ANN. To satisfy the tight constraints on memory size imposed by embedded systems and to avoid overfitting, we employ ANNs with a small number of trainable parameters. The primary purpose of our framework is the recognition of low-quality images of identity documents with complex backgrounds and a variety of languages and fonts. We demonstrate that our solution shows high recognition accuracy on natural datasets even being trained on purely synthetic data. We use MIDV-500 and Census 1961 Project datasets for text line recognition. The proposed method considerably surpasses the algorithmic method implemented in Tesseract 3.05, the LSTM method (Tesseract 4.00), and unpublished method used in the ABBYY FineReader 15 system. Also, our framework is faster than other compared solutions. We show the language-independence of our segmenter with the experiment with Cyrillic, Armenian, and Chinese text lines.

INDEX TERMS Text recognition, artificial neural networks, character segmentation, character recognition, machine learning.

I. INTRODUCTION

Smartphones, tablet computers, and other mobile devices gain more and more popularity each day. Applications for such devices include government and commercial services that often require entering data from printed documents. Yet the text entry on modern touch-based keyboards is error-prone and time-consuming [1], [2]. Thus, several solutions appeared in recent years [3]–[6] for optical text recognition in images that are captured using mobile devices. These systems can be classified into two groups: client-server solutions, which transfer images to a "cloud" and require internet connection, and "on the device" methods that perform the recognition process without data transmission.

The associate editor coordinating the review of this manuscript and approving it for publication was Ah Hwee Tan.

Recognition of identity documents is a specific case since they contain sensitive personal information, and any application should guarantee the security of personal data. Consequently, it seems reasonable to say that the "on the device" methods fit better for their processing than the "cloud" solutions since the former present fewer security issues. The tight constraints on computational power and memory size imposed by the embedded and mobile systems limit the resources available and render many approaches unusable. Also, any noticeable delay while executing a smartphone application can affect user experience. Moreover, such a system faces a lot of challenges. Unlike invoices and other official papers, identity documents often have complex backgrounds. Besides, they usually have specific surfaces. For example, ID documents can be laminated, and, consequently, highlights can appear in images captured using

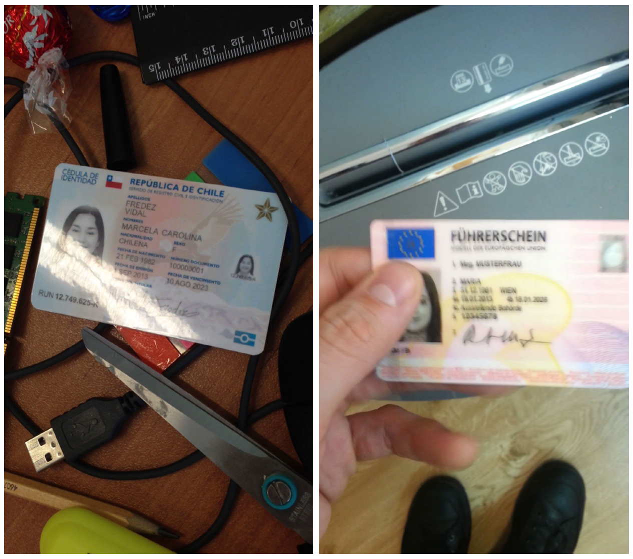


FIGURE 1. Example images from MIDV-500.

a smartphone. In addition, the process of image acquisition tends to introduce many distortions [7]–[9], which make methods designed for scanned documents images unsuitable for camera-captured ones [10]. Fig. 1 presents samples of camera-captured ID images from MIDV-500 dataset [11], which demonstrates typical input images for a recognition application on a mobile device. As a result, any competent approach to the embedded recognition of IDs should employ rather sophisticated methods and be efficient in the system with stringent resources.

The process of ID recognition can be divided into a number of steps, such as document identification and location, zone extraction and rectification, per-field segmentation, field recognition, language model postprocessing, and result acquisition [6]. In our study, we consider the field recognition step, which includes text line detection and recognition (Fig. 2), and focus on the text line recognition part. As identity documents have the predefined structure, the algorithms for straight line identification [12] are used for text line detection in a text field image. The text line vertical position can be specified in different ways that are common for both printed and handwritten text lines [13]. In our experiments, we suppose that baseline and cap line (Fig. 3) approximate coordinates are found beforehand, and we get them as input.

In this paper, we present a method for text line recognition that can be used as a part of an on-the-premises recognition system for various printed documents including but not limited to identity cards, passports, and driving licences.

II. RELATED WORK

Most of the modern text line recognition techniques can be divided into two large groups: the ones with explicit per-character segmentation followed by recognition and the end-to-end solutions. In sections II-A and II-B we briefly describe the existing approaches from both groups.

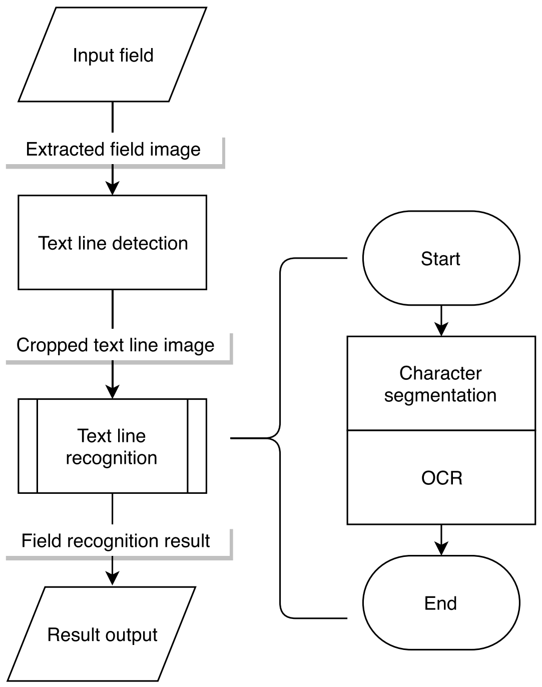


FIGURE 2. Field recognition scheme.

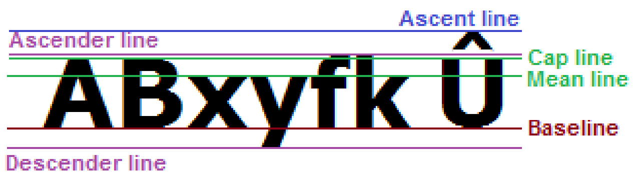


FIGURE 3. Representation of a text line.

A. RECOGNITION WITH EXPLICIT SEGMENTATION

Per-character line segmentation is a process of finding bounding rectangles of characters and is one of the essential problems of text recognition [12], [14]. Segmentation can be applied to binarized or grayscale images. However, in the case of camera-captured images, binarization can have a drastic effect of introducing background noise [7]. After the segmentation is performed, a classifier is used for OCR. The classification approaches vary greatly from pattern matching to the most modern architectures of neural networks. The majority of the recent studies focus on the segmentation part of these methods as it is believed to be more difficult than recognition [15].

Two main approaches to the per-character segmentation exist: projection analysis and connected components processing. The typical problem faced by the standard methods is the segmentation of either connected symbols or individual symbols consisting of several separate primitives, especially in the context of complex backgrounds and camera-specific distortions. Most methods include various heuristics about possible glyphs [5], [16], [17], which allow cutting or merging components of specific shapes. To overcome all the difficulties, over-segmentation methods are developed.

These methods try either to evaluate different segmentation paths and select the best one [7] or to restore correct segmentation from an over-segmented line using dynamic programming [18]. But even these methods use prior information about the text. For instance, a method [7] is fine-tuned for text printed with a monospaced font and method [18] does not consider the existence of characters consisting of several primitives. Remarkably, methods based on over-segmentation often employ artificial neural networks (ANN) for result improvement in the following way: a classifier, e.g., ANN, is trained to distinguish a character from its over-segmented part and the classification results are taken into account for the choice of the best path [19], [20]. To our best knowledge, it is the only way ANNs are used in explicit per-character segmentation. It is an omission as the ANNs, especially fully convolutional networks (FCN), demonstrate the state-of-the-art results in the contiguous task of semantic segmentation [21], [22].

The current situation with per-character segmentation can be regarded as the one with the omnifont character recognition before the rise of deep learning. While it is possible to perform optical character recognition (OCR) exploiting features that are pre-defined in the algorithms without any ANNs, such approaches are usually outperformed with modern ANN-based methods. However, most of the modern segmentation methods work in the same way–with the pre-defined language-dependable features. And as ANNs have already influenced OCR, they can significantly improve per-character segmentation.

B. END-TO-END RECOGNITION

Techniques without explicit segmentation, or end-to-end recognition, appeared due to the claim that the segmentation algorithms are highly error-prone for distorted images [23]. Modern segmentation-free text recognition methods are predominantly represented by various ANN-based methods and include the sliding-window classifiers [24] and the ones based on recurrent neural networks (RNN) [25], [26]. Nowadays RNNs demonstrate state-of-the-art results in the text-in-the-wild recognition problem [27], [28], and are used in various OCR engines, e.g., Tesseract 4.00 [29]. The main advantage of RNNs is their ability to handle sequences of characters, but it can make them ineffective for identity documents recognition as these documents contain, firstly, fields without language model (document numbers) and, secondly, rare spellings of popular names which can be "corrected" by the classifier [30]. The other variant of segmentation-free recognition is the utilization of per-word classifiers, i.e., one class stands not for a character, but a whole word [31]. This approach faces two main problems: the impossibility of recognition of document fields without language model and an enormous number of the resulting classes.

C. ANNS IN "ON THE DEVICE" OCR SOLUTIONS

OCR is the core part of any text recognition framework. The results of ICDAR2015 competition on smartphone document capture and OCR [32] demonstrate that OCR in camera-captured images remains an open problem even for documents with simple backgrounds. At the same time, the extensive use of mobile devices makes OCR a "must-have" application on modern gadgets [33]. Since the introduction of a convolutional neural network (CNN) LeNet-5 and its results on the MNIST dataset [34], the OCR task is usually solved with various ANNs that demonstrate state-of-the-art results on public datasets for object classification [35]–[39]. However, to be usable, ANNs employed in "on the device" software should satisfy the tight constraints on computational power and memory size [40]. In particular, it is essential for multi-language applications that require several classifiers. Consequently, many deep architectures need re-thinking for such applications [41]. Moreover, such methods as model ensembling employed in [35] could become quite resource-intensive. Thus, a number of different approaches to ANN modifications were introduced in recent years. Some of them suggest the creation of efficient light-weight architectures and modification of the state-of-the-art ones [42]–[48], others introduce solutions with 8-bit fixed point or binarized weights [49]–[51]. It should be mentioned that a few hardware-aware and hardware-adapted solutions exist [49], [52], but we are not interested in them in our study as we need a solution for off-the-shelf systems. Most of the suggested architectures have hundreds of thousands of weights. As the recent studies claim that the majority of the modern networks have the excessive capacity and are prone to overfitting [53], we intend to use feed-forward networks that have a small number of trainable parameters and are suitable for embedded systems. The light-weight ANN architectures that we propose in our framework are based on the previous papers of the authors. For example, in [54]–[56] we showed the capabilities of the light-weight architecture in OCR problems, in [58] we employed a light-weight neural network for both optical font recognition and OCR, and in [59] the ability of a light-weight ANN to detect vanishing points was demonstrated.

D. TEXT LINE RECOGNITION TOOLSETS

Nowadays, a number of text recognition systems are available. Some of them are available as the on-premises solutions, and some provide the client-server interfaces. In our study, we limit the set of considered systems to the two in the most common use: Tesseract OCR and ABBYY FineReader. Tesseract OCR is a free OCR engine, and ABBYY FineReader is a commercial product.

1) TESSERACT OCR

Tesseract OCR [60] is one of the most popular open-source OCR engines [61], widely employed by both developers and users. According to [32], [61]–[64], this system is commonly used within the community as a baseline method for recognition quality evaluation. Nowadays, Tesseract is being developed by Google. To assess our method, we compare its recognition results with those of Tesseract OCR of versions 4.00 and 3.05. Tesseract OCR 4.00 was released on October 29, 2018, and employs an ANN with LSTM blocks. This ANN contains 7.8×10^5 weights and has been trained on a large amount of purely synthetic data [29]. While the whole training data is not available online, the provided examples and the generation process description show that Tesseract 4.00 can be used for reference [65]. We additionally provide the results of the previous version, i.e., Tesseract OCR 3.05, as, firstly, it allows comparison with earlier studies, and, secondly, it demonstrated competitive results with other methods [61], [62].

2) ABBYY FINEREADER

ABBYY FineReader [66] is a state-of-the-art commercial OCR application [61] that is used in both scientific studies [7], [32], [62], [65] and business. In our experiments, we used the latest ABBYY FineReader 15. FineReader is considered one of the best solutions available on the market, but its source code or the employed algorithms are not published yet. Judging by the information provided on the official website, we can presume that machine learning algorithms with language models are used for text recognition.

III. DATASETS

Datasets is a widely used mean for method evaluation and comparison of various methods. The importance of public datasets in studies cannot be overestimated as they allow not to reproduce previously published methods for comparison, and, what is more, allow the comparison with the state-of-the-art commercial systems that do not provide enough information for method reproduction.

Throughout our study, we want to evaluate both the proposed method and its separate components. As a result, we need three types of test datasets: one suitable for the segmenter evaluation, one for the classifier evaluation, and one for the full framework evaluation. To evaluate character classification, we used the famous MNIST dataset (Sec. III-A). The dataset for the segmentation evaluation is more tricky, as to directly assess it we need per-character segmentation ground truth. To our best knowledge, no public dataset provides such a ground truth. Thus, we use synthetic data (Sec. III-D) for segmenter evaluation.

As for the overall framework evaluation, we use two recently introduced public datasets: MIDV-500 (Sec. III-B) and the test part of the 1961 Census for England and Wales from [61] (Sec. III-C). These datasets are of great interest to us as they provide text line segmentation ground truth, i.e., they allow evaluating text line recognition independently from document location and text line segmentation methods. In fact, such datasets are a rarity within the modern scientific community. Firstly, private datasets are widely used [7], [62], [65] for text line recognition evaluation. Secondly, public datasets are often designed for overall document recognition systems. The vivid example of this problem is a widely used SmartDoc-2015 dataset [32]. This dataset contains camera-captured images of documents and text ground truth. Yet, if one wants to use it for the evaluation of text line recognition quality, they have to employ exterior methods for document detection and text line segmentation.

Since our method employs ANNs, we also need training data. We mainly use synthetic training data as they allow us to achieve sufficient accuracy and to get samples for various languages and scripts. Also, synthetic data provides the necessary font and background diversity without laborious and time-consuming data preparation. Thus, we employ synthetic training data (Sec. III-D) in all experiments except the one with MNIST. We use the MNIST default training sample to compare our classifier with the previously published ones without training data influence.

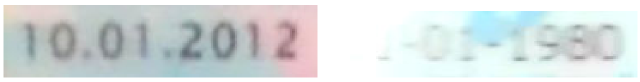
A. MNIST

MNIST consists of 70000 images of handwritten digits, 60000 of which form the training set, and the rest 10000 are treated as the test set. The state-of-the-art result reported for MNIST is the 0.23% error rate for non-ensemble classifiers. We use this dataset to assess the suggested light-weight architecture of the per-character classifier: we train our per-character classifier on the training set of MNIST and then calculate the classification error-rate of the acquired classifier on the MNIST test set.

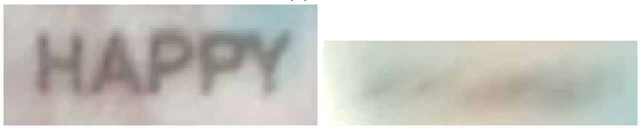
B. MIDV-500

MIDV-500 [11] is of particular interest for us as it contains images of ID samples. Each document is captured in five different conditions with two mobile devices (Apple iPhone 5 and Samsung Galaxy S3). Resulting snapshots contain complex background and various distortions that are typical for images acquired with smartphones [11] (Fig. 1). Text lines in MIDV-500 are printed with multiple proportional and monospaced fonts.

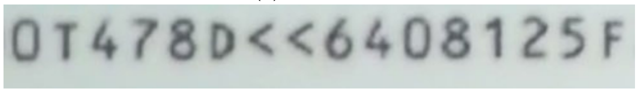
We preprocess MIDV-500 using the provided ground truth to 1) select snapshots with fully visible documents as the dataset contains some images with documents partially hidden off-screen; 2) perform projective rectification; 3) extract images of separate text lines and divide these lines into four groups which are distinguished in the ground truth: "Dates" – numeric dates, "Latin names" – names and surnames printed with Latin letters without diacritical marks, "MRZ" – machine-readable zones [67], and "Docnum" – document numbers. The exact statistics on the acquired text lines are given in Table 1. We do not filter out the blurred, highlighted, or otherwise degraded images. Fig. 4 shows the examples for each type of text line. Initially, the total number of character classes in the test dataset was equal to 70 and included digits, punctuation symbols "(),.-/< " and both capital and small Latin letters. As we train case-insensitive classifiers and also unify the letter "O" and the digit "0", the resulting number of classes became 43.



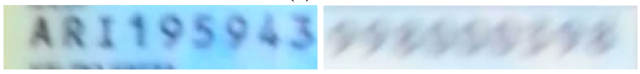
(a) "Dates"



(b) "Latin names"



(c) "MRZ"



(d) "Docnum"

FIGURE 4. **Examples of text lines from MIDV-500.**

TABLE 1. **The distribution of text line images in MIDV-500.**

Field type	Images	Symbols
Dates	17735	176142
Latin names	15257	131067
MRZ	5096	220992
Docnum	9329	88294

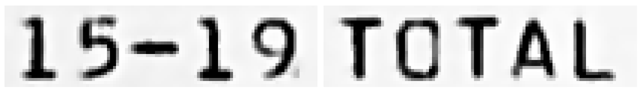


FIGURE 5. **Examples of text lines from the 1961 Census sample.**

C. THE 1961 CENSUS FOR ENGLAND AND WALES SUBSAMPLE

The employed subsample of the 1961 Census for England is presented in [61] and is available for download from PRImA website [68]. It contains scanned images of documents printed with one font. This test sample is suitable for our method, as the authors provide the geometrical ground truth as well as the textual one. Thus, to perform experiments, we used the provided ground truth to acquire separate images of text lines. The text lines in this set contain capital Latin letters, punctuation symbols "`&,.-/`" and digits. Fig. 5 presents the text line images from this dataset.

D. SYNTHETIC DATASETS

We utilize no data from MIDV-500 for training CNNs. Instead, we use a method described in [69] to generate a synthetic training dataset. This method allows us to create an unlimited number of images with projective transformation, motion blur, and ink degradation. As we want our model to be as language-independent as possible, we employ no dictionaries in the generation process. To preserve the generalization capability of our ANNs, we do not choose any specific fonts. We utilize 600 different fonts that are available on GoogleFonts [70] and backgrounds that were acquired

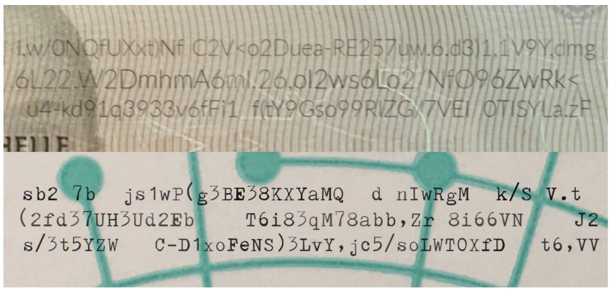


FIGURE 6. **Example generated images from training dataset.**

from images of various documents. Examples of the synthetic images are shown in Fig. 6. As we train case-insensitive recognition network and also do not distinguish the letter "`O`" and the digit "`0`" the number of samples per character differs, but the number of examples per class stays approximately equal. For instance, each class consists of approximately 25000 samples, but for class "`1`" all samples contain the same character and class "`O`" contains "`O`", "`o`" and "`0`" in equal proportions. This training dataset is available at `ftp://smartengines.com/2cnntrain/`.

What is more, we use synthetic data as both training and test sets to evaluate the per-character segmentation part of the proposed method. To be exact, we generate three training datasets (Chinese characters, Cyrillic characters, and a mixture of Cyrillic, Latin, and Armenian characters) and three test datasets (Chinese characters, Cyrillic characters, and Armenian characters). We select these writing systems for the test datasets as, firstly, they are rather different, and, secondly, two of them are in the top five most widespread scripts around the world, and the third one is a rare writing system which is used mostly in Armenia and, consequently, is rarely considered in OCR systems. We do not use the Latin characters in test sets as they are similar to the Cyrillic ones.

Each dataset contains ground truth for each image. The annotation was performed automatically by the data generation engine. The ground truth is given in JSON format and is organized as follows (Fig. 7): 1) one JSON dictionary stands for one text line; 2) "`line_rect`" is the bounding box of the text line, "`cuts_x`" stands for the cuts between the letters, i.e. ideal segmentation results, "`start_x`" and "`end_x`" outlines the exact coordinates of character end points in horizontal direction, "`values`" are the Unicode characters codes in a decimal numerical system, "`let_blines`" stands for upper and lower ordinates of each character, and "`blines`" represents the lines from Fig. 3 except for the ascender line.

IV. A METHOD FOR TEXT LINE RECOGNITION

In a classic Heuristic Over-Segmentation method, a segmentation graph is formed from the vertical projection (projection on the horizontal axis) P [34], [71]. To solve the task, one looks for the best path from the start node to the end node. What is crucial, the cuts from P are usually selected once, before the graph construction. It means that if the cut

```
[
    {
        "line_rect": [75, 55, 570, 40],
        "cuts_x": [87, 108, 131, 154, 185],
        "start_x": [77, 87, 108, 131, 155],
        "end_x": [83, 108, 129, 153, 183],
        "valid": true,
        "values": [59, 99, 103, 51, 67],
        "let_blines": [[63,93], [62,87],
            [62, 95],[55, 87], [55, 87]],
        "blines": [55, 56, 64, 86, 95]
    },
    {
        //next line description
    },
    ...
]
```

FIGURE 7. Generated image ground truth in JSON format.

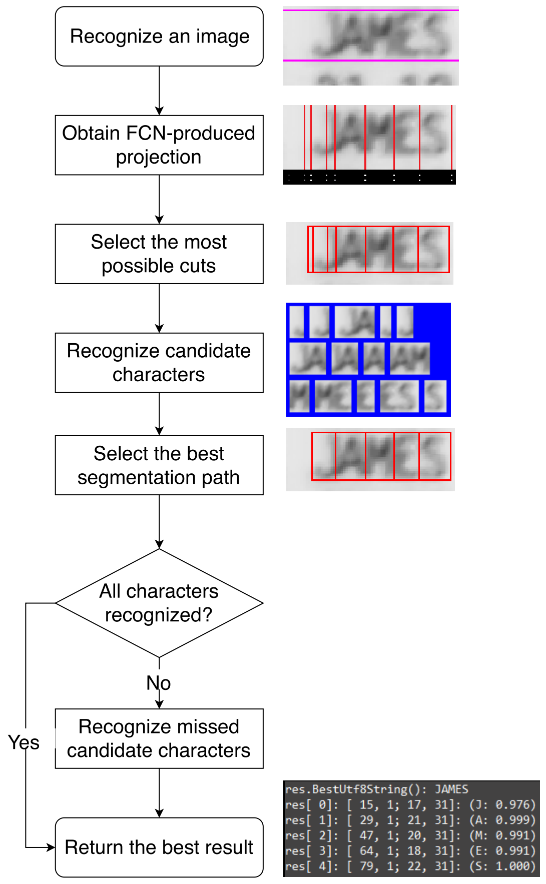
was initially missed, it could not be restored by the method. Besides, the start and end nodes should be determined beforehand. Such limitations seem to be presumptuous for any practical use in the case of images captured with a mobile device.

To overcome the drawbacks of the classical approach, we propose a method based on two ANNs, namely a segmenter and a classifier, and dynamic programming. To be exact, we employ an FCN NN_{segm} to build P and a CNN NN_{class} for candidate characters recognition. NN_{segm} and NN_{class} are described in Sec. IV-A and Sec. IV-B in detail. Fig. 8 presents the flowchart of our algorithm:

1) Crop an image using baseline and cap line, and scale the resulting image to a predefined height.
2) Apply NN_{segm} and obtain P.
3) Build P_1 as a non-maximum suppression of P.
4) Build pairs of non-zero points from P_1 with the distance from the predetermined interval $fd \in [fd_{min}, fd_{max}]$.
5) Classify the preliminary candidate character images with NN_{class}.
6) Employ dynamic programming to build the segmentation path optimizing the sum of the cuts scores from P and confidences of the corresponding candidate characters, if any.
7) Classify characters at the positions from the built path that were not processed in steps (3)-(4), if any.

Firstly, we apply NN_{segm} to the cropped text line image scaled to the predefined height and obtain the projection P that represents the score of a cut for every column in the image. Fig. 9 provides the result obtained with the segmentation network for the text line image from Fig. 8.

Then, we build the initial subset P_1 of the most probable cuts from P, selecting the local maxima of P. Based on P_1, we classify preliminary candidate characters with NN_{class}. The preliminary candidate characters can be mutually exclusive, i.e., their bounding boxes can overlap. It should be mentioned that we do not classify all the candidate characters.


Recognize an image
Obtain FCN-produced projection
Select the most possible cuts
Recognize candidate characters
Select the best segmentation path
All characters recognized?
No
Yes
Recognize missed candidate characters
Return the best result

```
res.BestUtf8String(): JAMES
res[ 0]: [ 15, 1; 17, 31]: (J: 0.976)
res[ 1]: [ 29, 1; 21, 31]: (A: 0.999)
res[ 2]: [ 47, 1; 20, 31]: (M: 0.991)
res[ 3]: [ 64, 1; 18, 31]: (E: 0.991)
res[ 4]: [ 79, 1; 22, 31]: (S: 1.000)
```

FIGURE 8. Text line recognition scheme.

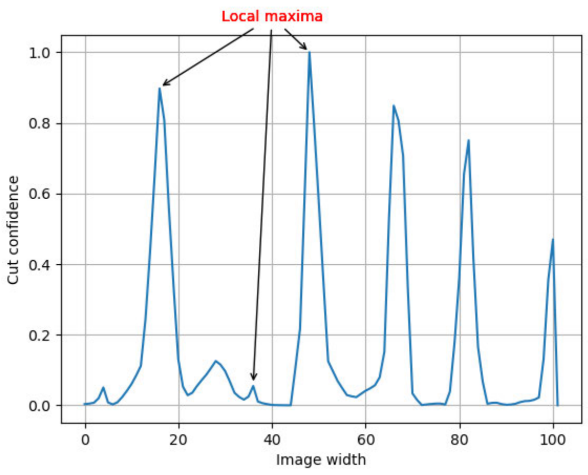


FIGURE 9. Cuts confidence obtained with NN_{segm}.

We sort the candidate characters according to the sum of their forming cuts from P_1, and then select m most probable ones, where m is the quotient of the ratio between input image width and the mean possible character width from fd. After the candidates classification, we employ dynamic programming to select the best segmentation path. What is crucial, the absence of some necessary cuts in P_1 does not inevitably lead to the incorrect result as we allow the discontinuity of the best segmentation path in relation to preliminary candidate characters. To be more exact, when we build a segmentation path, we look into P, not P_1, for cuts scores and consider all possible candidate characters. We evaluate each segmentation path as the mean of the scores of its candidate characters. The score s of one candidate character is calculated in (1):

$$s = 0.5 \times (P(x_l) + P(x_r)) + c \tag{1}$$

where x_l and x_r are the cuts used to form the candidate, c is the confidence of the first alternative obtained with NN_{class} for preliminary classified characters, or 0, if the candidate was not classified. After the best segmentation path is chosen, we check it for the missed character candidates and classify them.

Speaking about candidate characters, we obtain their images not from the cropped image, but from the original input one. The reason is that at this stage we analyze the horizontal projections of individual character images and adjust their vertical borders to handle diacritics, descenders, and ascenders. Also, we allow setting a restriction to the $width/height$ ratio of the selected candidate character images as an input parameter to our framework. This restriction is set in the form of possible minimal and maximal $width/height$ ratio and is used to forbid impossibly narrow or wide symbols.

Fig. 10 explains how we acquire the result with dynamic programming in more detail. In Fig. 10, we calculate current image width x by the horizontal axis and the number of characters k by the vertical one. As we mentioned before, we preset the interval of possible character widths $fd \in [fd_{min}, fd_{max}]$. Thus, we calculate the maximal number of characters k_{max} as:

$$k_{max} = \frac{w_{img}}{fd_{min}}, \tag{2}$$

where w_{img} denotes the width of the input image. As we limit the possible width of the character, we do not need to calculate all the possible states in dynamic programming. In Fig. 10, we denote the possible states as green cells. Bright green cells denote the possible initial states, and dark green ones denote the final states. In our solution, we explicitly restrict the possible initial and final states. For any initial state $x \in [0, fd_{max}]$ and for any final state $x \in [w_{img} - fd_{max}, w_{img}]$.

As the initial state of our value function we set the scores of all the possible first characters to:

$$dp(x, 0) = 0.5 \times P(x). \tag{3}$$

$$dp(x, k) = 0.5 \times P(x) + dp(x^*, k-1) + C(x^*, x),$$
$$x^* = \operatorname*{argmax}_{x_{pr} \in [x - fd_{max}, x - fd_{min}]} dp(x_{pr}, k-1), \tag{4}$$

where x^* denotes the previous cut coordinate, $C(x^*, x)$ is the confidence of the first alternative obtained with NN_{class} for candidate character between x^* and x, or 0 if the candidate was not classified. Fig. 11 shows part of the best segmentation path selection process for the image from Fig. 8.

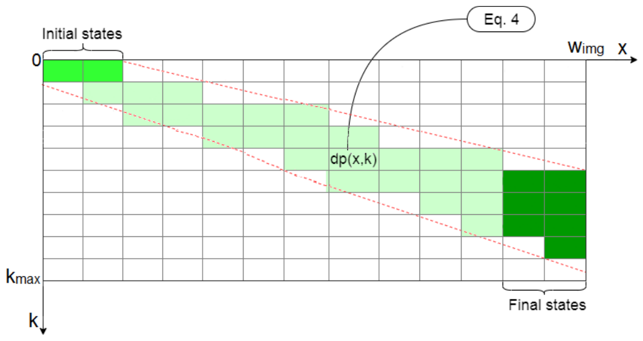


FIGURE 10. Dynamic programming employment.

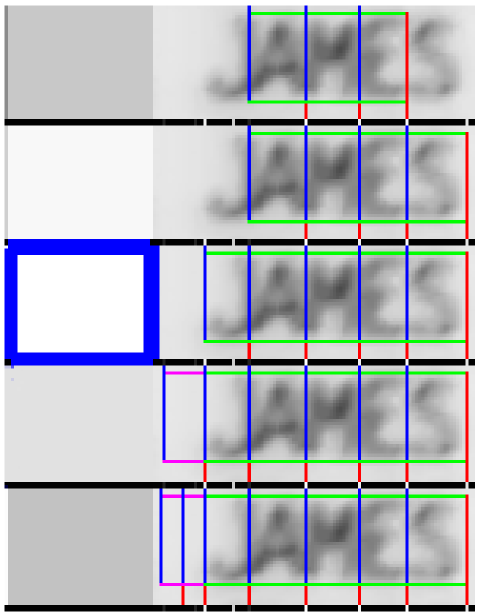


FIGURE 11. Selection of the best segmentation path.

We use two separate ANNs instead of an end-to-end solution for two main reasons. The first one is that the character recognition quality depends on the correspondence between the alphabets of the classifier and text lines. The second one is that segmentation can be considered as a language-independent problem for a large number of languages, and, to some extent, as the one independent of the writing system as well. Due to the same reasons, we do not utilize segmentation methods based on specific knowledge about characters in the considered text lines. Conversely, the recognition ANN will be error-prone if we merge several alphabets. For example, the resulting set of characters from all the Latin-script alphabets would contain about ten "I" letters with different diacritical marks; composition English and Russian alphabets would include many identical symbols, so we would be bound to perform a postprocessing stage to obtain the result.

A. SEGMENTER

As it was already mentioned, we employ the FCN to compute a vertical projection of the image. Using the information from its receptive field, the segmentation neural network decides

TABLE 2. Architecture of the segmentation FCN.

Layers			
#	Type	Activation function	Parameters
1	conv	ReLU	6 filters 5×5, stride 2×1, no padding
2	conv	ReLU	6 filters 5×5, stride 1×2, padding 0×2
3	conv	ReLU	6 filters 3×7, stride 1×1, padding 0×3
4	conv	ReLU	6 filters 5×5, stride 1×1, padding 0×2
5	conv	ReLU	6 filters 3×3, stride 1×1, padding 0×1
6	conv	ReLU	6 filters 5×5, stride 1×2, padding 0×2
7	conv	ReLU	6 filters 5×2, stride 1×1, no padding
8	conv	ReLU	6 filters 5×5, stride 1×2, padding 0×1
9	conv	Tanh	6 filters 5×3, stride 1×1, padding 0×1
10	conv	1-RBF	1 filter 3×3, stride 1×1, no padding

whether there is a cut at a given position or not. As a result, we acquire a projection containing the network's scores for a cut at each column in the image. In the inference stage, this neural network can be applied to an image of arbitrary width since all its layers are size-independent. Table 2 describes the architecture of the FCN, which we use in our experiments. The approximate number of trainable parameters of this FCN is 5.7×10^3. To train the network, we use grayscale images of fixed size 131×33 and the ground truth in the form of 131×1 images that represent an ideal projection – zero-filled image with one-filled points at the places of correct cuts. All the images and ground truth are taken from the dataset described in Sec. III-D. The training was performed with the minimization of Euclidean distance between the calculated and ideal projections.

B. CLASSIFIER

To recognize the characters, we employ the ANN with light-weight architecture, which is provided in Table 3. This CNN takes grayscale images of fixed size 15×19 as an input. The number of trainable parameters of this CNN depends on the size of the alphabet $|A|$ as its last layer is a fully-connected one. For a classification task with the alphabet size of 30, it is approximately 3.4×10^4.

V. EXPERIMENTAL RESULTS

A. CLASSIFIER EVALUATION

To begin with, we experiment with MNIST to evaluate the error rate of the recognition network with architecture from Table 3 with $|A| = 10$. We do not employ any additional training data or data preprocessing but use online augmentation [55]. Table 4 provides the results obtained by the suggested CNN and those of previously published studies, and also shows the number of trainable parameters in the applied ANNs. We get an error rate of 0.25% against 0.23% demonstrated by the best non-ensemble model and 0.14% - by the ensemble one. In Table 4, we present the best results that can be found in the published papers. Reference [72] provides the results known before 2013. We also provide the human performance error rate – ≈0.20% – to emphasize that the majority of the classifiers presented in Table 4 are almost equal to human recognition ability. We claim that this

TABLE 3. Architecture of the classification CNN.

Layers			
#	Type	Activation function	Parameters
1	conv	ReLU	8 filters 3×3, stride 1×1, no padding
2	conv	ReLU	8 filters 3×3, stride 1×1, padding 1×1
3	conv	ReLU	16 filters 3×3, stride 2×2, padding 1×1
4	conv	ReLU	16 filters 3×3, stride 1×1, padding 1×1
5	conv	ReLU	16 filters 3×3, stride 1×1, padding 1×1
6	conv	Tanh	32 filters 3×3, stride 2×2, padding 1×1
7	conv	Tanh	32 filters 3×3, stride 1×1, padding 1×1
8	conv	Tanh	32 filters 3×3, stride 1×1, padding 1×1
9	conv	Tanh	8 filters 3×3, stride 1×1, padding 1×1
10	fully connected with softmax		$\lvert A\rvert$ neurons

TABLE 4. MNIST recognition results.

Model	Test error rate (%)	Parameters $\times10^6$	Voting ANNs
State-of-the-art R.F. Alvear-Sandoval et al. [73]	0.14	35.4	15
Human performance Y. LeCun et al. [74]	≈0.20	N/A	N/A
L. Wan et al. [35]	0.21	≈ 20.0	5
I. Sato et al. [75]	0.23	0.099	1
D. Cireşan et al. [76]	0.23	2.660	35
J.-R. Chang et al. [77]	0.24	0.259	1
Our CNN	**0.25**	**0.034**	1
S.Sabour et al. [78]	0.25	8.2	1
M. Liang et al. [79]	0.31	0.670	1
S.Sabour et al. [78]	0.35	6.8	1

result on the MNIST database proves the applicability of such a light-weight ANN to the OCR problem.

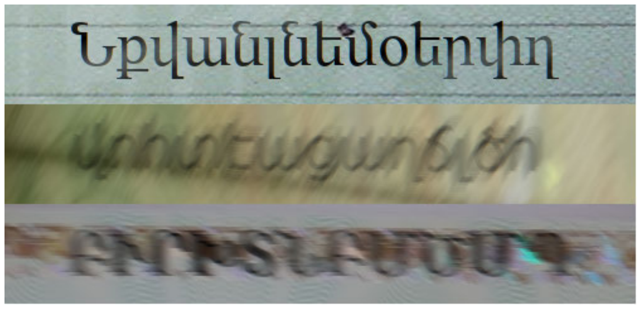
B. SEGMENTER EVALUATION

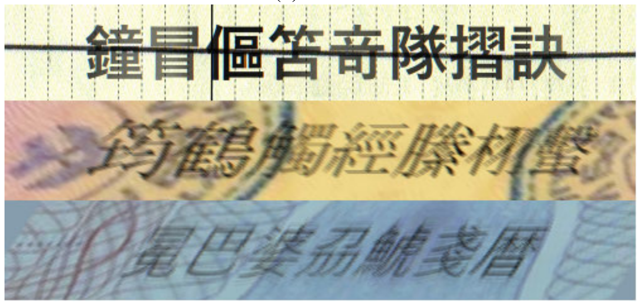
In the second stage, we focus on the proposed segmentation network and its claimed language-independence. It should be emphasized that by the language-independent method we mean that our segmentation network can be built with a lot of languages and writing systems taken into account. To verify the proposed method, we use the datasets that are described in Sec. III-D. Fig. 12 provides example test images. The images in Fig. 12 are intended to show both the diversity of characters and distortions in the test set. This diversity prevents us from using classic image processing methods.

To evaluate only the segmentation, we exclude the second summand from eq. 1. We calculate the segmentation error rate as:

$$SER = \frac{\sum_{i=1}^{L_{total}} CM(P_i, l_{i_{cuts}})}{L_{total}}, \tag{5}$$

where L_{total} denotes the total number of lines, $l_{i_{cuts}}$ is the ground truth cuts, P_i is the computed projection, and $CM(P_i, l_{i_{cuts}})$ means the result of the discrete two-sample Cramér-von Mises test [56]. Table 5 provides the segmentation results, the value of eq. 5 if no "cuts" were found and the worst-case value of eq. 5 for each test set. It should be emphasized that the Cramér-von Mises test depends on


(a) Armenian


(b) Chinese

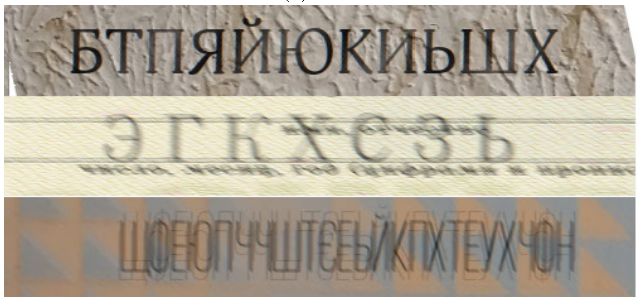
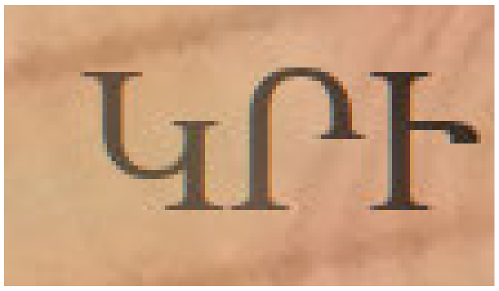
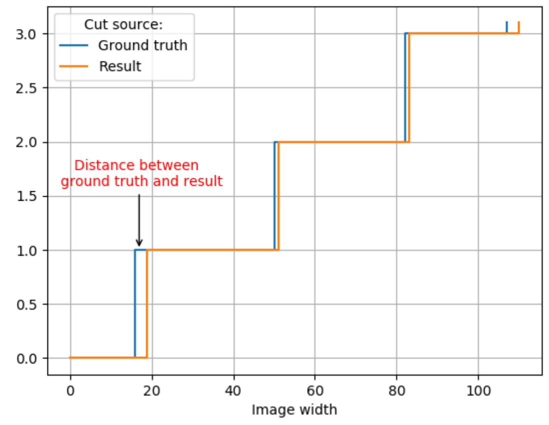

(c) Cyrillic

FIGURE 12. Example generated images from segmentation test sets.

TABLE 5. Text lines segmentation results.

		Test dataset		
		Chinese	Cyrillic	Armenian
Training dataset	Chinese	0.20	1.32	1.09
	Cyrillic	0.20	0.79	0.70
	Mixture	0.18	0.79	0.64
No cuts		4.54	7.45	6.88
Worst possible score		245.61	244.19	244.62


(a) Part of a text line image



(b) Cumulative sum of the number of cuts

FIGURE 13. Difference between the ideal and obtained cuts.

the number of "cuts" in the text line, on the width of the characters, and on the distance between characters. Consequently, the results of one segmentation algorithm on several datasets cannot be directly compared. In the case of Table 5, it means that we can compare the values in one column, but to compare the values in one row, we should take the "No segmentation" results into account. In Fig. 13, we provide an example of such a noncomparability. Fig. 13a presents the beginning of the text line, and 13b shows the cumulative sums of the number of cuts from the ground truth and segmentation result that are used in the Cramér-von Mises test. As one can see, even for the first cut, there is a difference between the ideal and the obtained cuts. Such a small difference is allowable, but it will affect the resulting segmentation rate. The effect caused by such differences accrues with the length of the text line. In Table 5, the segmentation error for the "Cyrillic" test set is higher as the average "Cyrillic" text line is longer than the "Chinese" and "Armenian" ones. As one can see from the results (Table 5), the "mixture" segmentation network outperforms the other ones even on the Chinese hieroglyphs that were not presented in its training set. Also, the "Cyrillic" network demonstrates the results which are equal to the ones of the "Chinese" network on the Chinese set, but not vice versa. The reason is, on the one hand, the similarity of forms of the most hieroglyphs so that FCN mostly "saw" straight line segments and almost no arcs, and, on the other hand, the variety of geometric primitives presented in the Cyrillic script. We hypothesize that results on the Armenian set implicitly confirm this assumption as the "Chinese" network demonstrates worse results than the "Cyrillic" one.

C. EVALUATION OF TEXT LINE RECOGNITION FRAMEWORK

All components of our framework consider a single-channel image as an input. Thus, when an RGB image comes in, we average the three intensity values of each pixel. Also, we resize the image if it is necessary for the component, i.e., we resize the image to the height of 33 pixels preserving the width for the segmenter (Sec. IV-A) and we resize each candidate character image to 15×19 pixels as required by the classifier (Sec. IV-B). No other image pre-processing is performed.

1) MIDV-500 RESULTS

Finally, we experiment with MIDV-500 to evaluate the complete text line recognition framework. We train the segmentation network (Table 2) and the classification network (Table 3) employing traditional single precision (32-bit) computations. As the evaluation measure for the acquired results, we calculate the per-character recognition rate *PCR*.

$$PCR = 1 - \frac{\sum_{i=1}^{L_{total}} \min(lev(l_{i_{ideal}}, l_{i_{recog}}), len(l_{i_{ideal}}))}{\sum_{i=1}^{L_{total}} len(l_{i_{ideal}})} \quad (6)$$

where L_{total} denotes the total number of lines, $len(l_{i_{ideal}})$ is the length of the i-th line and $lev(l_{i_{ideal}}, l_{i_{recog}})$ stands for the Levenshtein distance between the recognized text and the ground truth. Table 6 provides the results of the proposed approach in comparison with Tesseract 4.00, Tesseract 3.05, and ABBYY FineReader 15. In our experiments, we used the modes of both versions of Tesseract OCR and FineReader that allow recognition of Latin characters, digits, and punctuation marks. In Tesseract OCR, we also disabled postprocessing with dictionaries as, firstly, some of the text lines do not have any language model, and, secondly, our framework does not employ any postprocessing. To evaluate the time consumption of the proposed algorithm in comparison with Tesseract, we measure the total time (in seconds) necessary for processing all images for each field on the personal computer on CPU AMD Ryzen 7 1700 on a single thread (Table 7). We do not provide a comparison with the execution time of ABBYY FineReader, as it is an application with a graphical user interface. Also, we do not provide time comparison with methods from Table 4 as they are designed for individual character recognition, not for the text line recognition.

A significant problem faced by the recognition systems for the camera-captured images is errors of document boundaries detection. It results in incorrectly rectified field images and distorted characters. To estimate the robustness of our method to such errors, we perform an experiment to model document detection errors employing the method described in [11]. For each image available in MIDV-500, we modify the document quadrangle by the addition of normally distributed noise to each vertex. We change each coordinate by five pixels at most as we want, firstly, to preserve text visibility and, secondly, to avoid highly distorted images that are unsuitable for Tesseract 4.00. Then we acquire the text line images as we did for the original quadrangles and perform the text line recognition process for the resulting image. Table 8 provides the absolute difference between the results from Table 6 and the ones of this experiment.

According to the experimental results, our method outperforms the latest Tesseract 4.00 on all types of the text lines in both accuracy and speed. The proposed method is at least 2.07 times faster than Tesseract 4.00. Moreover, in all fields our method yields substantially fewer errors than both Tesseract 4.00 and ABBYY FineReader 15 even though it does not employ any language model postprocessing. The results on the "Docnum" images are of particular interest as these text lines, on the one hand, contain unstructured data that are unlikely to be represented in the training dataset and, on the other hand, are one of the most important fields of the document. Also, our method mostly produces errors because of its case insensitivity and the independent recognition of each symbol. To be exact, its most frequent inaccuracy is mistaking the capital "`I`" for the small "`l`" and vice versa, e.g., we obtain "`MARlA`" instead of "`MARIA`". Such inaccuracies could be fixed during the postprocessing stage with language models. As for Tesseract 3.05, which was added due to its popularity in earlier studies, it is substantially surpassed by both the proposed method and Tesseract 4.00. The correspondence between the results of Tesseract 3.05 and ABBYY FineReader 15 complies with the previously published results. As for the mentioned speed advantage of our method, we suppose the main reason for it is that we employ two ANNs with approximately 4.0×10^4 weights in total while Tesseract 4.00 employs ANN with 7.8×10^5 weights. Judging by results in Table 8, the proposed framework is more robust than the end-to-end solution employed in Tesseract 4.00. Remarkably, the most severe decrease in Tesseract 4.00 PCR happens in the "Latin names" text lines, where it demonstrated the result closest to the proposed method. The possible reason for such an outcome is that the isolated recognition errors affect the end-to-end solution more seriously than they influence our method with independent per-character recognition.

TABLE 6. Text lines recognition results of the proposed method, Tesseract 4.00, Tesseract 3.05 and ABBYY FineReader 15 for MIDV-500.

Text line type	Proposed method	Tesseract 4.00	Tesseract 3.05	FineReader 15
	PCR × 100%			
Latin names	**79.04**	75.44	37.29	55.76
Dates	**84.59**	57.80	41.85	56.67
MRZ	**92.98**	47.94	58.52	74.11
Docnum	**80.06**	41.83	27.27	57.11

TABLE 7. Total time required for all images recognition for each field in MIDV-500.

Text line type	Total time (seconds)		
	Proposed method	Tesseract 4.00	Tesseract 3.05
Latin names	**112.697**	304.269	714.159
Dates	**121.755**	371.914	696.674
MRZ	**233.179**	586.808	731.757
Docnum	**110.067**	227.961	446.367

TABLE 8. Difference between the original results and recognition results with noisy document image rectification (MIDV-500).

Text line type	Proposed method	Tesseract 4.00	Tesseract 3.05	FineReader 15
	PCR × 100% decrease			
Latin names	**1.65**	4.32	3.82	2.12
Dates	**1.87**	3.56	9.65	3.10
MRZ	**1.02**	1.30	1.03	4.27
Docnum	**2.14**	2.88	3.09	2.42

2) THE 1961 CENSUS FOR ENGLAND AND WALES SUBSAMPLE RESULTS

The second experiment that we perform to evaluate the whole framework is the one with The 1961 Census for England and Wales subsample. We use the same segmentation network but retrain the classifier as the previous one does not contain "&" and "," in its alphabet. The training data is acquired in the same way as described in Sec. III-D. The other difference between this experiment and the one with MIDV-500 is character recognition accuracy calculation. The previously published results for this sample are calculated with PRImA TextEval [80]. This tool uses the so-called University of Nevada measure that is based on Levenshtein distance to compute character recognition accuracy. As TextEval is freely available, we use it to get recognition results comparable with previously presented ones. Table 9 provides the experimental results of the various methods and systems on the 1961 Census subsample. It should be mentioned that the authors of [61] retrain Tesseract Engine with the data obtained from the documents of the same 1961 Census for England and Wales. In other words, their training sample consists of character images printed with a specific font while we do not adjust the synthetic data generator for this particular document.

TABLE 9. Text line recognition results for the 1961 Census sample.

Method / System	OCR accuracy (%)
Tesseract 3.04	90.08
Tesseract 4.00	91.13
FineReader Engine 11 [61]	88.40
Proposed method	**96.69**
Retrained Tesseract [61]	94.51
Retrained Tesseract with dictionaries [61]	95.40

The acquired results demonstrate that our language-independent model surpasses methods that employ both the implicit language models (LSTM in Tesseract OCR 4.00) and the explicit ones (dictionaries in [61]). What is more, the most frequent error of our method is mistaking "O" for "C" when the former are distorted as demonstrated in Fig. 14. Fig. 15 provides the text lines images with incorrectly recognized "O". Our method is prone to these errors as it does not employ any dictionaries and could not understand that "TOTAL" is much more probable than "TCTAL". Even the simplest dictionary-based postprocessing could significantly improve the accuracy of our method as this error takes 35.10% of all character recognition errors committed by our method.

3) EXTENDED LATIN AND NON-LATIN TEXT LINE EXAMPLES

To check the language-independence of the framework on the real data, we applied it to several images from MIDV-500 containing non-Latin characters and Latin characters

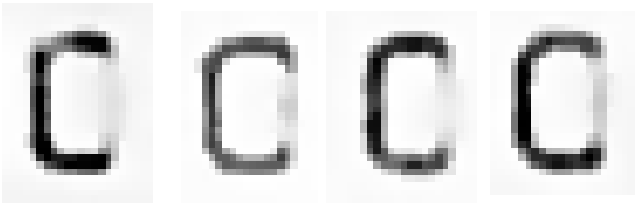
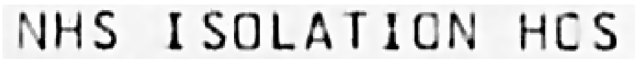
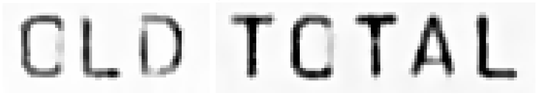


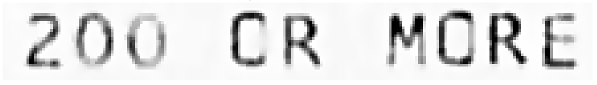
FIGURE 14. Distorted Os from the 1961 Census sample.



(a) Recognized as NHS ISOLATION HCS



(b) Recognized as CLD and TCTAL



(c) Recognized as 200 CR MORE

FIGURE 15. Text line images with incorrectly recognized O.

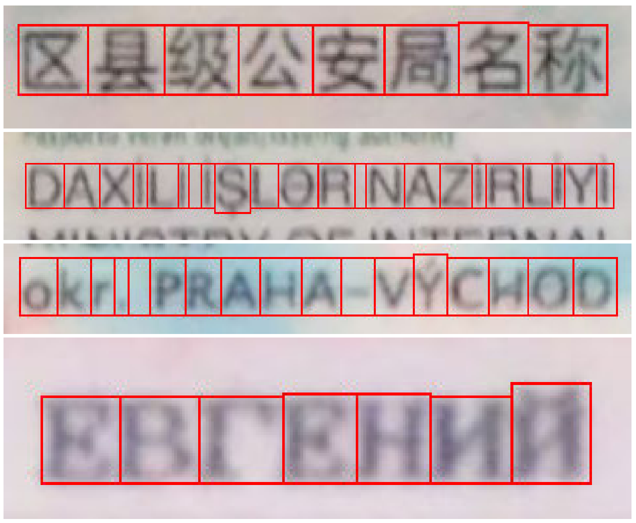


FIGURE 16. Segmentation results for different writing systems.

with diacritical marks. To do these experiments, we also trained the classifiers for four different alphabets without re-training the segmenter. We employed the procedure described in III-D to generate training data for the classifiers. Fig. 16 provides the results acquired for Chinese, Azerbaijani, Czech, and Russian characters. According to the results, we claim that the segmentation network can be made language-independent if trained with enough data. It should be mentioned that for recognition of the Chinese characters, we employ an ANN described in [57]. This ANN is similar to our classifier but provides the embeddings of the input images. Thus, the number of neurons in the last layer does not become enormously large, as it represents the dimensionality of embedding space, not the alphabet size. To be exact, the employed architecture proposed in [57] contains 7.7×10^4

weights, which is twice the size of our basic classifier, but it is light-weight in the case of Chinese characters.

VI. CONCLUSION

In this paper, we present our method for text line recognition that employs two ANNs interconnected by the dynamic programming algorithm. The primary motivation for the proposed approach is to solve the per character segmentation task as the language-independent one. As data acquisition is an obstacle for training ANNs for different languages, we utilize only the synthetic training data in our central experiment.

We provide a comparison of the recognition accuracy results of our method, LSTM-based Tesseract 4.00, the algorithmic method from Tesseract 3.05, and ABBYY FineReader 15 on the public dataset for the camera-captured ID recognition MIDV-500. The acquired results show that our framework is essentially better than ABBYY FineReader 15 and both versions of Tesseract OCR. Also, we provide the results of our method on the 1961 Census of England and Wales Project dataset. We achieve the highest recognition accuracy in comparison with previously published results of several LSTM-based and algorithmic methods. Moreover, we would like to mention that the recognition accuracy of our framework in images with names and other dictionary words can be improved significantly by using the language model postprocessing. We demonstrate the transferability of the segmentation network to different scripts if connected with the appropriate recognition ANN.

To justify the applicability of the suggested light-weight classifier, we experiment with the classic MNIST dataset and acquire the results comparable with the state-of-the-art ones. To examine the segmentation method, we show that the segmentation networks trained on the data with different alphabets perform almost equally on different datasets. We employ synthetic data for this experiment as we need per-character segmentation ground truth.

To conclude, our framework demonstrates the powerful capabilities of employing the FCNs for text line segmentation and of using extremely light-weight ANNs for camera-captured image recognition.

REFERENCES

[1] S. Azenkot and S. Zhai, "Touch behavior with different postures on soft smartphone keyboards," in *Proc. MobileHCI*, San Francisco, CA, USA, 2012, pp. 251–260.

[2] S. Ruan, J. O. Wobbrock, K. Liou, A. Ng, and J. Landay, "Comparing speech and keyboard text entry for short messages in two languages on touchscreen phones," 2018, *arXiv:1608.07323*. [Online]. Available: https://arxiv.org/abs/1608.07323

[3] Y. Wang, Y. Sun, and C. Liu, "Layout and perspective distortion independent recognition of captured chinese document image," in *Proc. 14th IAPR Int. Conf. Document Anal. Recognit. (ICDAR)*, Nov. 2017, pp. 591–596.

[4] M.M. Luqman, pp. Gomez–Krämer, and JM. Ogier, "Mobile phone camera-based video scanning of paper documents," in *Camera-Based Document Analysis and Recognition* (Lecture Notes in Computer Science), vol. 8357. Cham, Switzerland: Springer, 2013.

[5] B. Q. L. Mai, T. H. Huynh, and A. D. Doan, "An independent character recognizer for distantly acquired mobile phone text images," in *Proc. Int. Conf. Adv. Technol. Commun. (ATC)*, Oct. 2016, pp. 85–90.

[6] K. Bulatov, V. V. Arlazarov, T. Chernov, O. Slavin, and D. Nikolaev, "Smart IDReader: Document recognition in video stream," in *Proc. 14th IAPR Int. Conf. Document Anal. Recognit. (ICDAR)*, Nov. 2017, pp. 39–44.

[7] F. Jia, C. Shi, Y. Wang, C. Wang, and B. Xiao, "Grayscale-projection based optimal character segmentation for camera-captured faint text recognition," in *Proc. 14th IAPR Int. Conf. Document Anal. Recognit. (ICDAR)*, Nov. 2017, pp. 1301–1306.

[8] N. Nayef, M. M. Luqman, S. Prum, S. Eskenazi, J. Chazalon, and J. M. Ogier, "SmartDoc-QA: A dataset for quality assessment of smartphone captured document images-single and multiple distortions," in *Proc. ICDAR*, Nancy, France, Aug. 2015, pp. 1231–1235.

[9] K. Bulatov, "Selecting optimal strategy for combining per-frame character recognition results in video stream," *J. Inf. Technol. Comput. Syst.*, no. 3, pp. 45–55, 2017.

[10] Y. Wang, Y. Sun, and C. Liu, "Layout and perspective distortion independent recognition of captured chinese document image," in *Proc. 14th IAPR Int. Conf. Document Anal. Recognit. (ICDAR)*, Nov. 2017, pp. 591–596.

[11] V. V. Arlazarov, K. Bulatov, T. Chernov, and V. L. Arlazarov, "A dataset for identity documents analysis and recognition on mobile devices in video stream," *Comput. Opt.*, vol. 43, no. 5, pp. 818–824, 2019, doi: 10.18287/2412-6179-2019-43-5-818-824.

[12] S. Eskenazi, P. Gomez-Krämer, and J.-M. Ogier, "A comprehensive survey of mostly textual document segmentation algorithms since 2008," *Pattern Recognit.*, vol. 64, pp. 1–14, Apr. 2017.

[13] G. Renton, C. Chatelain, S. Adam, C. Kermorvant, and T. Paquet, "Handwritten text line segmentation using fully convolutional network," in *Proc. 14th IAPR Int. Conf. Document Anal. Recognit. (ICDAR)*, vol. 1. Kyoto, Japan, Nov. 2017, pp. 5–9.

[14] P. Sahare and S. B. Dhok, "Multilingual character segmentation and recognition schemes for Indian document images," *IEEE Access*, vol. 6, pp. 10603–10617, 2018.

[15] P. Lu, L. Shan, J. Li, and X. Liu, "A new segmentation method for connected characters in CAPTCHA," in *Proc. Int. Conf. Control, Autom. Inf. Sci. (ICCAIS)*, Oct. 2015, pp. 128–131.

[16] P. Shivakumara, S. Bhowmick, B. Su, C. L. Tan, and U. Pal, "A new gradient based character segmentation method for video text recognition," in *Proc. Int. Conf. Document Anal. Recognit.*, Sep. 2011, pp. 126–130.

[17] U. Pal, P. Pratim Roy, N. Tripathy, and J. Lladós, "Multi-oriented Bangla and Devnagari text recognition," *Pattern Recognit.*, vol. 43, no. 12, pp. 4124–4136, Dec. 2010.

[18] P. Pratim Roy, U. Pal, J. Lladós, and M. Delalandre, "Multi-oriented touching text character segmentation in graphical documents using dynamic programming," *Pattern Recognit.*, vol. 45, no. 5, pp. 1972–1983, May 2012.

[19] T. Saba and A. Rehman, "Effects of artificially intelligent tools on pattern recognition," *Int. J. Mach. Learn. Cyber.*, vol. 4, no. 2, pp. 155–162, Apr. 2013.

[20] R. Hussain, H. Gao, and R. A. Shaikh, "Segmentation of connected characters in text-based CAPTCHAs for intelligent character recognition," *Multimedia Tools Appl.*, vol. 76, no. 24, pp. 25547–25561, Dec. 2017.

[21] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 39, Jun. 2015, pp. 640–651.

[22] G. Nanfack, A. Elhassouny, and R. O. H. Thami, "Squeeze-SegNet: A new fast deep convolutional neural network for semantic segmentation," *Proc. SPIE*, vol. 10696, Apr. 2018, Art. no. 106962O.

[23] C. H. Chen and J. L. DeCurtins, "Word recognition in a segmentation-free approach to OCR," in *Proc. 2nd Int. Conf. Document Anal. Recognit. (ICDAR)*, Tsukuba, Japan, Dec. 2002.

[24] H. El Bahi and A. Zatni, "Text recognition in document images obtained by a smartphone based on deep convolutional and recurrent neural network," *Multimedia Tools Appl.*, vol. 78, no. 18, pp. 26453–26481, Sep. 2019.

[25] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," 2015, *arXiv:1507.05717*. [Online]. Available: https://arxiv.org/abs/1507.05717

[26] T. Chernov, D. Ilin, P. Bezmaternykh, I. Faradzhev, and S. Karpenko, "Research of segmentation methods for images of document textual blocks based on the structural analysis and machine learning," *RFBR J.*, vol. 92, no. 4, pp. 55–71, 2016.

[27] B. Su and S. Lu, "Accurate scene text recognition based on recurrent neural network," in *Computer Vision* (Lecture Notes in Computer Science), vol. 9003, D.Cremers, I. Reid, H. Saito, and M. H. Yang, Eds. Cham, Switzerland: Springer, 2014.

[28] B. Su and S. Lu, "Accurate recognition of words in scenes without character segmentation using recurrent neural network," *Pattern Recognit.*, vol. 63, pp. 397–405, Mar. 2017.

[29] R. Smith. (2016). *Tesseract Blends Old and New OCR Technology*. [Online]. Available: https://github.com/tesseract-ocr/docs/tree/master/das_tutorial2016.

[30] A. Ul-Hasan and T. M. Breuel, "Can we build language-independent OCR using LSTM networks?" in *Proc. of 4th Int. Workshop Multilingual (OCR-MOCR)*, New York, NY, USA, 2013, p. 9.

[31] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Synthetic data and artificial neural networks for natural scene text recognition," 2014, *arXiv:1406.2227*.

[32] J. Burie, J. Chazalon, M. Coustaty, S. Eskenazi, M. M. Luqman, M. Mehri, N. Nayef, J. Ogier, S. Prum, and M. Rusinol, "ICDAR2015 competition on smartphone document capture and OCR (SmartDoc)," in *Proc. 2015 13th Int. Conf. Document Anal. Recognit. (ICDAR)*, Nancy, France, Aug. 2015, pp. 1161–1165.

[33] S. Paul, S. Saha, S. Basu, P. K. Saha, and M. Nasipuri, "Text localization in camera captured images using fuzzy distance transform based adaptive stroke filter," *Multimedia Tools Appl.*, vol. 78, no. 13, pp. 18017–18036, Jul. 2019.

[34] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[35] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proc. 30th Int. Conf. Mach. Learn. (ICML)*, vol. 28. Atlanta, GA, USA, May 2013, pp. 1058–1066.

[36] B. Graham, "Fractional max-pooling," 2015, *arXiv:1412.6071*. [Online]. Available: https://arxiv.org/abs/1412.6071

[37] DA. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," 2015, *arXiv:1511.07289*. [Online]. Available: https://arxiv.org/abs/1511.07289

[38] J. Zhao, M. Mathieu, R. Goroshin, and Y. LeCun, "Stacked what-where auto-encoders," 2015, *arXiv:1506.02351*. [Online]. Available: https://arxiv.org/abs/1506.02351

[39] C.-Y. Lee, P. W. Gallagher, and Z. Tu, "Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree," 2015, *arXiv:1509.08985*. [Online]. Available: https://arxiv.org/abs/1509.08985

[40] Z. Luo, A, Small, L. Dugan, and S. Lane, "Cloud Chaser: Real time deep learning computer vision on low computing power devices," *Proc. SPIE*, vol. 11041, Mar. 2019, Art. no. 110412Q.

[41] D. Kang, D. Kang, J. Kang, S. Yoo, and S. Ha, "Joint optimization of speed, accuracy, and energy for embedded image recognition systems," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2018, pp. 715–720.

[42] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. 2018 IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 6848–6856.

[43] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*. [Online]. Available: https://arxiv.org/abs/1704.04861

[44] S. H. Hasanpour, M. Rouhani, M. Fayyaz, M. Sabokrou, and E. Adeli, "Towards principled design of deep convolutional networks: Introducing SimpNet," 2018, *arXiv:1802.06205*. [Online]. Available: https://arxiv.org/abs/1802.06205

[45] S. H. Hasanpour, M. Rouhani, M. Fayyaz, and M. Sabokrou, "Lets keep it simple, Using simple architectures to outperform deeper and more complex architectures," 2018, *arXiv:1608.06037*. [Online]. Available: https://arxiv.org/abs/1608.06037

[46] Y. Weng and C. Xia, "A new deep learning-based handwritten character recognition system on mobile computing devices," *Mobile Netw. Appl.*, to be published.

[47] S. Han, J. Tran, J. Pool, and W. J. Dally, "Learning both weights and connections for efficient neural network," in *Proc. NIPS*, Montreal, QC, Canada, 2015, pp. 1135–1143.

[48] C. Alippi, S. Disabato, and M. Roveri, "Moving convolutional neural networks to embedded systems: The AlexNet and VGG-16 case," in *Proc. 17th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw. (IPSN)*, Porto, Portugal, Apr. 2018, pp. 212–223.

[49] A. Ignatov, R. Timofte, W. Chou, K. Wang, M. Wu, T. Hartley, and L. Van Gool, "Ai benchmark: Running deep neural networks on Android smartphones," in *Proc. ECCV*, Munich, Germany, Sep. 2018, pp. 288–314.

[50] P. Gysel, J. Pimentel, M. Motamedi, and S. Ghiasi, "Ristretto: A framework for empirical study of resource-efficient inference in convolutional neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5784–5789, Nov. 2018.

[51] J. Kung, D. Zhang, G. van der Wal, S. Chai, and S. Mukhopadhyay, "Efficient object detection using embedded binarized neural networks," *J. Signal Process. Syst.*, vol. 90, no. 6, pp. 877–890, Jun. 2018.

[52] A. Gholami, K. Kwon, B. Wu, Z. Tai, X. Yue, P. Jin, S. Zhao, and K. Keutzer, "SqueezeNext: Hardware-aware neural network design," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Salt Lake City, UT, USA, Jun. 2018, pp. 1638–1647.

[53] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," 2017, *arXiv:1611.03530*. [Online]. Available: https://arxiv.org/abs/1611.03530

[54] A. N. Chirvonaya, A. E. Lynchenko, Y. S. Chernyshova, and A. V. Sheshkus, "Comparison of the classifying and similarity metric-based neural networks through the recognition of the filed," *Sensory Syst.*, vol. 33, no. 1, pp. 65–69, 2019, doi: 10.1134/S0235009219010049.

[55] A. V. Gayer, Y. S. Chernyshova, and A. V. Sheshkus, "Effective real-time augmentation of training dataset for the neural networks learning," *Proc. SPIE*, vol. 11041, Mar. 2019, Art. no. 110411I.

[56] A. V. Sheshkus, Y. S. Chernyshova, A. N. Chirvonaya, and D. P. Nikolaev, "New criteria for neural network encoder learning in the string segmentation problem," *Sensory Syst.*, vol. 33, no. 2, pp. 173–178, 2019.

[57] S. A. Ilyuhin, A. V. Sheshkus, and V. L. Arlazarov, "Recognition of images of korean characters using embedded networks," 2019, *arXiv:1911.04241*. [Online]. Available: https://arxiv.org/abs/1911.04241

[58] Y. S. Chernyshova, M. A. Aliev, E. S. Gushchanskaia, and A. V. Sheshkus, "Optical font recognition in smartphone-captured images and its applicability for ID forgery detection," *Proc. SPIE*, vol. 11041, Mar. 2019, Art. no. 110411J, doi: 10.1117/12.2522955.

[59] A. Sheshkus, A. Ingacheva, V. Arlazarov, and D. Nikolaev, "HoughNet: Neural network architecture for vanishing points detection," in *Proc. Int. Conf. Document Anal. Recognit. (ICDAR)*, Sep. 2019, pp. 844–849.

[60] (2018). *Tesseract OCR*. [Online]. Available: https://github.com/tesseract-ocr

[61] C. Clausner, A. Antonacopoulos, and S. Pletschacher, "Efficient and effective OCR engine training," *Int. J. Document Anal. Recognit.*, to be published.

[62] A. P. Tafti, A. Baghaie, M. Assefi, H. R. Arabnia, Z. Yu, and P. Peissig, "OCR as a service: An experimental evaluation of Google Docs OCR, Tesseract, ABBYY FineReader, and Transym," in *Proc. ISVC*, vol. 2016, Las Vegas, NV, USA, 2016, pp. 735–746.

[63] F. Asad, A. Ul-Hasan, F. Shafait, and A. Dengel, "High performance OCR for camera-captured blurred documents with LSTM networks," in *Proc. 12th IAPR Workshop Document Anal. Syst. (DAS)*, Santorini, Greece, Apr. 2016, pp. 7–12.

[64] M. Kišš, M. Hradiš, and O. Kodym, "Brno mobile OCR dataset," 2019, *arXiv:1907.01307*. [Online]. Available: https://arxiv.org/abs/1907.01307

[65] M. Namysl and I. Konya, "Efficient, lexicon-free OCR using deep learning," 2019, *CoRR*, abs/1906.01969, 2019.

[66] (2019). *ABBYY FineReader*. [Online]. Available: https://www.abbyy.com/en-eu/finereader/

[67] *International Civil Aviation Organization Doc 9303*. Accessed: Jan. 10, 2020. [Online]. Available: https://www.icao.int/publications/pages/publication.aspx?docnum=9303

[68] *Dataset for Paper Efficient and Effective OCR Engine Training*. Accessed: Jan. 10, 2020. [Online]. Available: https://www.primaresearch.org/datasets/TESSERACT_TRAINING.

[69] Y. Chernyshova, A. Gayer, and A. Sheshkus, "Generation method of synthetic training data for mobile OCR system," *Proc. SPIE*, vol. 10696, Apr. 2018, Art. no. 106962G.

[70] *GoogleFonts*. Accessed: Jan. 10, 2020. [Online]. Available: https://fonts.google.com/

[71] C. J. C. Burges, O. Matan, Y. Le Cun, J. S. Denker, L. D. Jackel, C. E. Stenard, C. R. Nohl, and J. I. Ben, "Shortest path segmentation: A method for training a neural network to recognize character strings," in *Proc. IJCNN Int. Joint Conf. Neural Netw.*, Jan. 2003, pp. 165–172.

[72] Y. LeCunn, C. Cortes, and C. C. J. Burges. *The MNIST Database of Handwritten Digits*. Accessed: Aug. 7, 2019. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[73] R. F. Alvear-Sandoval, J. L. Sancho-Gómez, and A. R. Figueiras-Vidal, "On improving CNNs performance: The case of MNIST," *Inf. Fusion*, vol. 52, pp. 106–109, Dec. 2019.
[74] Y. LeCun, L. D. Jackel, L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. A. Müller, E. Säckinger, P. Simard, and V. Vapnik, "Learning algorithms for classification: A comparison on handwritten digit recognition," in *Proc. ICNN*, Perth, WA, Australia, 1995, pp. 53–60.
[75] I. Sato, H. Nishimura, and K. Yokoi, "APAC: Augmented pattern classification with neural networks," 2015, *arXiv:1505.03229*. [Online]. Available: https://arxiv.org/abs/1505.03229
[76] D. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Providence, RI, USA, Jun. 2012, pp. 3642–3649.
[77] J.-R. Chang and Y.-S. Chen, "Batch-normalized maxout network in network," 2015, *arXiv:1511.02583*. [Online]. Available: https://arxiv.org/abs/1511.02583
[78] S. Sabour, N. Frosst, and J. E. Hinton, "Dynamic routing between capsules," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 3856–3866.
[79] M. Liang and X. Hu, "Recurrent convolutional neural network for object recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Boston, MA, USA, Jun. 2015, pp. 3367–3375.
[80] *PRImA Text Evaluation Tool*. Accessed: Jan. 10, 2020. [Online]. Available: http://www.primaresearch.org/tools/PerformanceEvaluation

YULIA S. CHERNYSHOVA received the B.S. degree in applied mathematics and the M.S. degree in applied computer science from the National University of Science and Technology MISIS, Moscow, Russia, in 2016 and 2018, respectively. She is currently pursuing the Ph.D. degree in computer science with the Federal Research Center Computer Science and Control of RAS, Moscow.

She is also with Smart Engines. Her research interests include training data synthesis, optical character recognition, and deep neural networks.

ALEXANDER V. SHESHKUS received the B.S. and M.S. degrees in applied physics and mathematics from the Moscow Institute of Physics and Technology, State University, Moscow, Russia, in 2009 and 2011, respectively.

He is currently the Head of the Machine Learning Department, Smart Engines, and a Researcher with the Federal Research Center Computer Science and Control of RAS. His research interests include deep neural networks, computer vision, and projective invariant image segmentation.

VLADIMIR V. ARLAZAROV received the Ph.D. degree in applied mathematics from the Moscow Institute of Steel and Alloys, in 1999.

He is currently an Associate Professor with the Moscow Institute of Physics and Technology, State University, Moscow, Russia, and a Senior Researcher with the Institute for Information Transmission Problems of RAS, Moscow. His research interests are pattern recognition and machine learning.