SerbOCR - Optical Character Recognizer of the Serbian Cyrillic Alphabet

Jovan Vještica, Teodora Đorđević, Suzana Stojković, *University of Niš, Faculty of Electronic Engineering*

Abstract—**This paper presents a technique for optical character recognition of machine-written Serbian Cyrillic text based on deep Convolutional Neural Networks. The described OCR system makes use of distinct preprocessing, character segmentation and character classification modules. For preprocessing, an adaptive gaussian threshold was used, while character segmentation used horizontal and vertical projection as well as connected component analysis. The classifier itself is a deep convolutional neural network. The entire system gave good results, with scanned documents yielding strings with a Levenshtein distance of less than 1% of the string length to the actual value present on the image.**

***Index Terms*—Optical Character Recognition; Neural Networks; Artificial Intelligence; Computer Vision; Image Processing**

I. INTRODUCTION

The oldest and most widespread mediums humans have used historically for storing information are visual mediums. As evidenced by a multitude of prehistoric cave paintings and ancient texts, humans have expressed their ideas, histories, and experiences in visual form since the dawn of humanity itself. Computer vision attempts to replicate the success of the human brain in recognizing optical patterns in order to decipher their meaning.

Optical character recognition, which in the modern context can be considered a subfield of computer vision, traces its roots to the first half of the 20th century, when the public was first introduced to practical machines capable of using optical signals as inputs. These machines, such as the Optophone [1], designed by Irish inventor Edmund Fournier d'Albe, were primarily created as aides for the visually impaired, thus allowing blind people to read regular print by producing a white noise of differing pitch.

In the post-war world of the second half of the 20th century, the market experienced a huge surge in both the availability and demand of consumer digital electronics. With the growing sophistication and power of these devices, as evidenced by Moore's law, the need to properly digitize "legacy" stores of information, such as books, newspapers, and letters, arose. In order to improve the searchability and usability of these document stores, OCR techniques have been developed and applied in order to translate matrices of pixels into arrays of characters.

Jovan Vještica is a student of Masters Studies at the Faculty of Electronic Engineering, University of Niš, Aleksandra Medvedeva 14, 18000 Niš, Serbia (e-mail: jovan.vjestica@elfak.rs).
Teodora Đorđević is with the Faculty of Electronic Engineering, University of Niš, Aleksandra Medvedeva 14, 18000 Niš, Serbia (e-mail: teodora.djordjevic@elfak.ni.ac.rs).
Suzana Stojković is with the Faculty of Electronic Engineering, University of Niš, Aleksandra Medvedeva 14, 18000 Niš, Serbia (e-mail: suzana.stojkovic@elfak.ni.ac.rs).

This paper presents a technique developed to analyze scanned documents written in the Serbian variant of the Cyrillic script. The approach that was developed contains following phases:

- image binarization by using an adaptive gaussian threshold,
- image rotation so that the text is parallel to the lower edge of the image,
- extracting the regions containing individual character based on connected component analysis and
- classification of image contents with a deep convolutional neural network.

II. RELATED WORK

Optical character recognition is a branch of computer vision that concerns itself with developing approaches to transforming a picture into characters. Typically, this process is broken up into multiple phases, each feeding its output into the input of the next [2]. At its most basic form, OCR involves two steps – generating rectangular regions which contain text and character extraction from those regions. This processing pipeline is often augmented in real applications by adding various preprocessing steps, mostly in the form of various filters. Postprocessing is also typically employed, primarily to increase the accuracy of the system for complex languages.

A. Binarization

Image binarization is one of the most important steps to consider when creating an OCR model. Image binarization attempts to map a matrix of pixels into a matrix of ones and zeros, with the intention of accentuating different structural components of individual letters by segmenting them from neighboring characters. The simplest approach of binarization is applying a fixed global threshold, but this approach typically gives poor results. Otsu's method [3] is vastly more popular and is widely used in OCR applications before and during text segmentation. Otsu's method determines a global threshold value that minimizes the foreground-background-class variances, which ensures good results when the image histogram has two distinct peaks. A new, iterative method based on Otsu's method has been proposed, which classifies pixels into three classes – foreground, background and to-be-determined, where the latter class is iteratively distributed into the foreground and background, respectively [3].

By applying an adaptive approach, not only was the pixel that is being processed taken into account, but also its immediate surroundings. The value being processed is either the mean value of the surrounding pixels or a gaussian-weighted sum [4].

A neural network-based approach has been proposed in [5] which uses the form of the histogram as well as a priori knowledge about the image as inputs to the general regression neural network that produces the final output.

B. Character segmentation

Character segmentation yields the locations of different characters presented on the image. Depending on the type of image being analyzed, different approaches may need to be applied. Identifying segmenting lines is important in document analysis, since it allows more efficient text segmentation. The most often used approach for line segmentation is horizontal projection, where different valleys indicate line breaks, typically represented as new line characters in documents. By analogy, an individual line segment can be split into word segments by applying vertical segmentation. The simplest and most direct approach to character segmentation is connected component analysis which can give satisfactory results when applied to preprocessed scanned documents [6]. Holistic methods exist which attempt to recognize words in their entirety thus evading the need for per-character segmentations [7].

C. Character classification

After properly segmenting the image, the final step is to apply the classification algorithm. Fundamentally, the goal of this step is to assign a class label based on an input matrix of fixed dimensions, where each class corresponds to each of the identifiable characters. Several methods can be applied here, including methods based on matrix-matching, feature engineering and neural networks [2]. A method based on neural network committees, which employ 7 different neural networks, has been applied very successfully to the MNIST dataset [8]. Additionally, there is no need to explicitly assign labels to characters – instead, a clustering algorithm may be applied to identify the different clusters, where each cluster represents a character. After postprocessing and human verification, the user can assign an explicit label in the form of an ASCII character to the cluster [9].

III. Image preprocessing and character segmentation

Since this paper deals with machine written documents that were scanned by a scanner, two key assumptions could be made – namely that all of the text lines present in the image were typed out at the same angle (this means that all the lines in the image were parallel) and that the lighting was consistent. The scanner itself, as well as the printing process of the document, can add noise to the scanned document. To denoise the image, preprocessing steps of edge extraction and thresholding are applied. Edge extraction is applied to remove the effects of background color and to better segment the individual characters [10]. The standard Canny edge detector from the OpenCV library is used, as it gave satisfactory results. The resulting edges are cleaned up and binarized by applying a global thresholding operation with a gaussian valued window.

After denoising the image, skew correction is applied. Skew correction is, in essence, a rotation of the image such that the text present on the rotated image spans horizontally instead of at an angle [11]. Since the image has been properly denoised and binarized, this is achieved by computing the minimal rotated rectangle that contains colored pixels. Next, an approach based on horizontal and vertical line projections is adopted to extract the different regions of interest that contain potential characters, while also applying connected component analysis to ensure that no characters are split.

To extract the regions containing characters, a horizontal projection to identify the text lines is applied. The begging of each line is denoted as a sharp decrease in the median value of the pixels along the scan line, while the end of a line is denoted as a sharp increase in the value. Similarly, the same method of splitting vertically in order to isolate individual characters can be applied. Since machine-written Serbian does not contain any ligatures nor does it contain any pair of letters that are connected, this approach works well. A potential problem that can arise in this phase of processing is the splitting of individual characters. To combat this, a variant of connected component analysis is applied by disallowing a vertical line to split a region if there are any colored pixels that have neighboring colored pixels in the next column. Additionally, once more a horizontal projection is applied on the newly extracted window. This is because lines will typically contain both capital and regular letters, which have varying heights. By choosing a fixed height for the entire line being analyzed, all regular letters would end up having a large blank upper portion, which would convey no information to the classifier later. Moreover, even regular letters in Serbian Cyrillic come in vastly different heights (for instance a and ђ), which becomes very evident in certain fonts.

As the last step in character extraction, the defined region is resized to a fixed 32 x 32 resolution. This is mandatory since the classifier can only classify 32 x 32 tensors. The resizing algorithm applied is based on bilinear interpolation, which is one of the commonly used algorithms for image resizing [12].

IV. Character classification in SerbOCR

After extracting all the regions that contain characters, classification is applied to transform the image, which is in this stage a 32 x 32 binary matrix, into a character. Several different classifier architectures were considered. Firstly, a standard multi layer perceptron network was used for classification, but it had severe drawbacks – namely, the loss function was never optimized fully, or the network would severely overfit, causing low precision on the testing data. After this, a shallow but wide convolutional network was considered, which gave acceptable results. Ultimately, a deeper network gave the best results. This architecture has found wide applications in image classification [13].

Before training the classifier, a dataset needed to be constructed. By applying the document segmentation algorithm, as described earlier, it was possible to isolate the individual characters present on the images that comprised the dataset. This ensured that both the format and position of the individual character in the region would be similar to what the classifier would encounter in a production environment, as well as having the added benefit of preserving the noise present on the initial image. This region was then classified using Tesseract, an open source OCR solution [14]. Based on the results of this classification process, the image was then stored onto the disk in the directory corresponding with the given class. In total 82 distinct classes were used, corresponding to the digits 0-9, all of the letters of the Serbian alphabet in both their regular and capital variants, as well as several special characters, such as parentheses, the exclamation mark, the question mark, and others. To increase the robustness of the classifier, all of the images in the dataset had a salt-and-pepper noise applied to them. This additional noise was meant to force the classifier to actually extract meaningful features from the images being analyzed. Taking into account the statistical properties of natural languages, it was expected that certain classes would be heavily favored, while others would conversely be neglected. Some classes, such as the vowels, were overrepresented in the dataset, while others were almost entirely absent. To combat this, data augmentation was applied to the underrepresented classes, such as applying paddings, slight stretches or small rotations and randomly cropping the resulting image to bring it to the needed size [15]. Each augmented data point used its own salt and pepper noise. Since the overrepresented classes had certain data points that were virtually identical to each other, a random sampling of these classes was applied. This dataset was then split into a training, testing and validation dataset, with an 8:1:1 ratio. In total, approximately 50 000 data points were used to train the classifier.

The most natural way of encoding the labels for the dataset was using the categorical encoding strategy (one-hot encoding). This approach unraveled the label column into 82 binary columns, where a value of 1 at index *i* denoted that the data point was a member of class *i*. Naturally, the new label vector can only feature a single 1 value, with the rest of the values being zeroes.

The classifier itself is a feed forward deep convolutional neural network. The network consists of four blocks, with an additional global maximum pooling layer and a fully connected layer at the end. Each neural network block consists of a two-dimensional convolution layer, followed by a batch normalization block. The entire network has been implemented using the keras library for interacting with the tensorflow backend.

The convolutional layers act as feature extractors, which operate by applying a learned convolution kernel to the data being processed. A standard square kernel with a dimension of 3 was used. Each convolutional layer uses the rectified linear unit activation function. By applying a padding operation and setting the convolution stride to 2 in both directions, a down sampling operation was applied by reducing the dimensionality of data by a factor of two in each dimension. Thus, there is no need to apply a maximum pooling operation, although this is common.

Batch normalization is a technique that can be used in convolutional neural networks to increase their training speed and stability and is often used to regularize deep convolutional models as a powerful alternative to the classic dropout approach [16]. A batch normalization layer is applied after each convolutional layer.

The penultimate layer in the neural network is a global maximum pooling layer, which reduces the dimensionality of the network to a single extracted feature vector. This vector serves as the input to the final layer of the neural network, which is a fully connected layer. This layer has 82 neurons, each corresponding to a potential class. The activation function of the final layer is the softmax activation function. This ensures that the 82 output values of the network can be interpreted as probabilities, specifically as the probability that image being classified belongs to the class associated with the neuron. The final value of the classification is inferred as the most probable class.

For training purposes, a loss function needed to be defined. Since at this stage this is essentially a multiclass classification problem, a natural candidate for the loss function is the categorical crossentropy function [17]. As the final performance metric, the validation loss is chosen instead of training loss, choosing to save the model version which minimized the validation loss rather than the training loss.

V. Evaluation

The implemented OCR solution was evaluated by using a series of documents issued by the Council of the University of Niš. These documents contained text with various formatting options, such as bold letters, italic letters, letters of different font size and letters of different fonts in general. In addition, all the documents are written in the Cyrillic script.

A. *Character segmentation*

Character segmentation was the part of the pipeline that introduced most errors. Typically, the errors were introduced by character pairs, that when printed together had no space between them and appeared as a single character. The most represented pair that fits this description was '**ст**'. Also, the errors were common with pairs: '**гл**', '**УЈ**', '**Уј**', '**ту**', '**ут**', '**уд**', etc.

TABLE I
Character Segmentation

Paragraph	Length	Estimated length	Success of segmentation
Paragraph1	228	202	88%
Paragraph2	169	154	91%
Paragraph3	351	322	92%

In Table I is shown the success of segmentation that was achieved. On average the segmentation algorithm detected less characters that were present. This was because of implemented connected component analysis, which ensured that a single character is never detected as many characters. The average accuracy of the segmentation algorithm was 90%.

B. *Classifier accuracy*

The classifier achieved good accuracy on the train and validation sets. The accuracy was over 96% on the validation set. The accuracy on the test set was lower owing to the fact that not all characters were equally present in the training set. For instance, the letter '**a**' was over a hundred times more present than the letter ‘**џ**’. The classifier mainly produced errors when it came to underrepresented letters and symbols in the training set such as ‘**џ**’, ‘**Ћ**’, ‘**Ж**’, ‘**ж**’, ’**;**’. Since these symbols are also statistically underrepresented in the Serbian language the overall classifier accuracy was not lowered substantially achieving a final accuracy of 95%.

C. Paragraph level analysis

The measure that was used to evaluate the performance of the system is the Levenshtein distance. The Levenshtein distance is a metric that measures how many edits need to be applied to a certain string in order to transform it into a different string [18]. A valid edit is considered an insertion, deletion, or substitution.

The following table shows the Levenshtein distance in relation to the string length on a per-paragraph basis for a selected document. Seeing as the Levenshtein distance depends primarily on distance, the following table displays both the paragraph length, expressed as the number of characters present in the paragraph, and the Levenshtein distance between the detected content of the paragraph and the actual content.

TABLE II
PER-PARAGRAPH LEVENSHTEIN DISTANCES

Length	CNN	Deep CNN
308	42	19
660	69	25
390	55	26
749	69	28

The first paragraph was very text dense, with not a small number of whitespace characters and special symbols. The second paragraph was structurally similar to the first, containing mostly plain text. Paragraph 3 contained a high number of punctuation characters. The fourth paragraph was like the first two.

The deep CNN consistently outperformed the shallow CNN, having on average fewer mistakes than the shallow CNN in every paragraph used for analysis. This was especially noticeable in examples 2 and 4, where the deep CNN outperformed the shallow CNN by a factor of 2.5.

As can be seen, most of the errors were induced by special symbols and whitespace characters. Special symbols were severely underrepresented in the training dataset, and after resizing, the classifier struggled to identify them correctly. White space characters get detected as part of the text segmentation process, and not the classifier, and thus improvement in the dataset would not improve this.

VI. CONCLUSION

This paper has outlined an OCR system created for the Serbian Cyrillic script. First, it is described how to perform the character segmentation from a scanned, machine written document. After this, the way the dataset used to train the classifier is generated, is described. Finally, the architecture of the classifier is shown and explained.

From all the above, it is shown that general purpose OCR techniques can be effectively specialized to be applied to machine written Cyrillic. There is still a lot of opportunity to improve the system described in this paper, primarily in the way special characters such as white spaces and others are handled. The analysis of cursive, handwritten Cyrillic also poses several new and interesting problems.

ACKNOWLEDGMENT

This work has been supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia.

REFERENCES

[1] Albe E. E. Fournier, L. O. Joseph, “On a type-reading Optophone”, Proceseedings of the Royal Society, vol. 90, no. 619, pp. 373-375, July 1914

[2] S. Singh, “Optical Character Recognition Techniques: A Survey”, *Journal of Emerging Trends in Computing and Information Sciences*, vol. 4, no. 6, pp. 2009-2015, June 2013

[3] N. Senthilkumaran, S. Vaithegi, “Image segmentation by using thresholding techniques for medical images”, *Computer Science & Engineering: An International Journal*, vol.6, no.1, February 2016

[4] S. Khurana, “Comparative Study on Threshold Techniques for Image Analysis”, *International Journal of Engineering Research & Technology*, vol. 4, no. 6, June 2015

[5] J. Lázaro, J. L. Martin, J. Arias, A. Astarloa, C. Cuadrado, “Neuro semantic thresholding using OCR software for high precision OCR applications”, *Image and Vision Computing*, vol. 28, no. 4, pp. 571–578, 2010

[6] A. Kaur, S. Baghila, S. Kumar, “Study of various character segmentation techniques for handwritten off-line cursive words: a review”, *International Journal of Advances in Science Engineering and Technology*, vol. 3, no. 3, pp. 154-158, July 2015

[7] R. G. Casey, E. Lecolinet, “A survey of methods and strategies in character segmentation”, *IEEE Transactions on Pattern Analysis and Machine Inteligence*, vol. 18, no. 7, pp. 690-706, August 1996

[8] D. C. Cires, U. Meier, L. M. Gambardella, J. Schmidhuber, “Convolutional Neural Network Committees For Handwritten Character Classification”, 2011 International Conference on Document Analysis and Recognition, Bejing, China, pp. 1135-1139, September 2011

[9] G. Vamvakas, B. Gatos, N. Stamatopoulos, S. J. Perantonis, “A Complete Optical Character Recognition Methodology for Historical Documents”, Document Analysis Systems, IAPR International Workshop, pp. 525-532, September 2008

[10] M. Zaharescu, I. C. Petrescu, “Edge detection in document analysis”, *Journal of Information Systems & Operations Management*, vol. 7, pp. 156-165, May 2013

[11] R. Ahmad, S. F. Rashid, M. Z. Afzal, M. Liwicki, A. Dengel, T. Breuel, “A novel skew detection and correction approach for scanned documents”, Document Analysis Systems, IAPR International Workshop, April 2016

[12] E. E. Danahy, S. S. Agaianb, K. A. Panettaa, “Algorithms for the resizing of binary and grayscale images using a logical transform”, *Image Processing: Algorithms and Systems V*, vol. 6497, February 2007

[13] W. Rawat, Z. Wang, “Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review”, *Neural Computation*, vol. 29, pp. 1-98, June 2017

[14] R. Smith, “An Overview of the Tesseract OCR Engine”, Ninth International Conference on Document Analysis and Recognition, vol. 2, pp. 629-633, September 2007

[15] A. Starynska, R. L. Easton, Jr. D. Messinger, “Methods of data augmentation for palimpsest character recognition with Deep Neural Network”, 4th International Workshop on Historical Document Imaging and Processing, pp. 54-58, November 2017

[16] E. Chai, M. Pilanci, B. Murmann, “Separating the Effects of Batch Normalization on CNN Training Speed and Stability Using Classical Adaptive Filter Theory”, February 2020

[17] Z. Zhang, M. R. Sabuncu, “Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels”, May 2018

[18] G. Cormode, S. Muthukrishnan, “The String Edit Distance Matching Problem with Moves”, *ACM Transactions on Algorithms*, vol. 3, no. 1, February 2007