Computer Science Journal of Moldova, vol.**29**, no.3(87), 2021

Towards a Font Classification Model for Romanian Cyrillic Documents

Tudor Bumbu

Abstract

This paper presents a solution on how to classify the fonts in the 17th century Romanian Cyrillic documents. This solution is based on a mix of unsupervised and supervised machine learning technics. The unsupervised process is the application of K-Means method to create the dataset with the fonts characters and their labels, whilst the supervised process is to train two different architectures of neural networks to classify these characters.

Keywords: old documents, OCR, font classification, neural networks, Romanian Cyrillic.

1 Introduction

Some Romanian Cyrillic documents of the 17th century require particular models at optical character recognition (OCR) because some printing houses used different character printing styles (fonts) than others [1].

The problem of identifying and classifying the font in a document printed in the 17th century can be formulated as follows: *Given a document X from 17th century printed in Cyrillic Romanian and a set N of OCR models trained on documents of the 17th and 18th century, choose the most appropriate model for X.*

A trivial solution is to recognize a sample (a page snippet) from document X using all models in N and basing on the results, to choose the model that gives the highest accuracy (best result). The time complexity of this solution does not fit our needs, as we have to load all OCR models, recognize the snippet and measure the accuracy for

each model separately. Model upload time and sample recognition can exceed 2 minutes depending on page size, and if we have 10 different models, we have to wait for approx. 20 minutes to find the the most appropriate model each time we want to recognize a document of this kind.

The proposed solution is to train a neural network with samples from several Romanian documents printed in the 17th century at different printing houses. A neural network is trained with a dataset consisting of tuples of *image character* and its *class* in order to be able to further classify a new sample.

In the next section we describe the selected document samples aiming at creating the dataset.

2 Dataset resources

The Romanian Cyrillic alphabet was used at printing houses in regions as *Iași, Bucharest, Târgoviște, Belgrade (Alba Iulia), Uniev (Cernăuți), Sas Sebeș, Snagov, Buzău.* Each of this region had at least one printing house in the 17th century.

The data set is created from 10 scanned books, selected from the digital library of Romania (http://digitool.bibnat.ro). In the selected books, two distinct sets of characters were observed. Therefore, the books were divided in two classes depending on their font style: one class of the books consisting of 13 pages was included in the set A, and the other class with 9 pages was included in the set B. Figure 1 shows two samples from each set A and B. Two main letters that differ in both samples are: *m* and *з* (*t* and *z* in Latin).

In the next section, we describe the main tasks in the process of creating the dataset: segmentation of text areas in the pages from A and B; detection of individual characters in text blocks; clustering the characters and forming the training and testing data set.

3 Creating the dataset

In the subsections below we describe tools for segmenting the regions of text (text blocks) in the selected pages, a method for identifying the

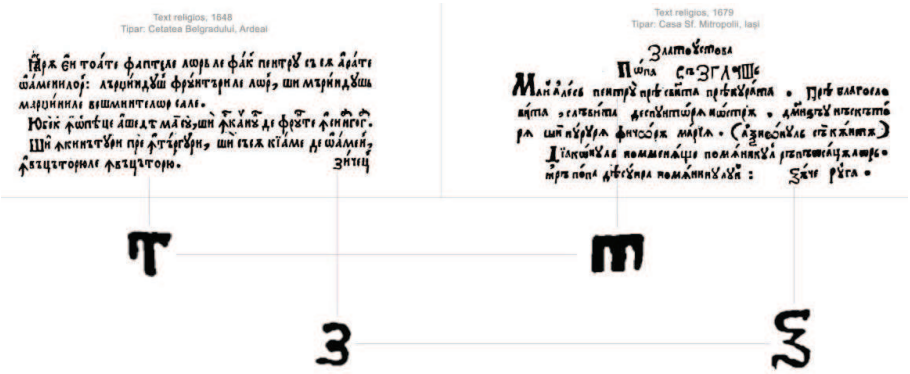

Figure 1. A sample from set A (on the left) and one sample from B

individual characters in a text block and approaches to group characters in similar groups in order to create a better dataset.

3.1 Segmentation of text blocks

From the pages prepared for the dataset we extracted fragments of text using *Detectron2* [2] segmentation tool trained on the *PrimaLayout* dataset [3].

In *PrimaLayout* dataset, the text portions are labeled with the label "TextRegion". We have extracted more than one block of text from every single page based on this label. For this reason, two blocks of text may contain the same characters, and similar examples of training and testing may appear in our dataset.

After segmentation, the text blocks were cut and placed in a list of text blocks. On average, 4 blocks of text were obtained for each of the 22 selected pages. These fragments are saved as images.

In the next subsection we identify and extract the characters from text blocks. Character set consists of *letters, punctuation marks, accents, outlines of tables, stain pixels,* or *page noise.*

3.2 Detecting individual characters of text blocks

We need to make sure that each image, independently of its source, is processed in such a way that the algorithm used to detect the

letters can find as many letters as possible. So, we converted all images to black and white. As a result, the images consisted only of black and white pixels. We optimized the image for letter detection using the *findCountours()* method within openCV library (https://docs.opencv.org/4.5.3/). We mapped the contour delimitation boxes to the original image to see what was actually detected. The result of this processing step is shown in Figure 2.

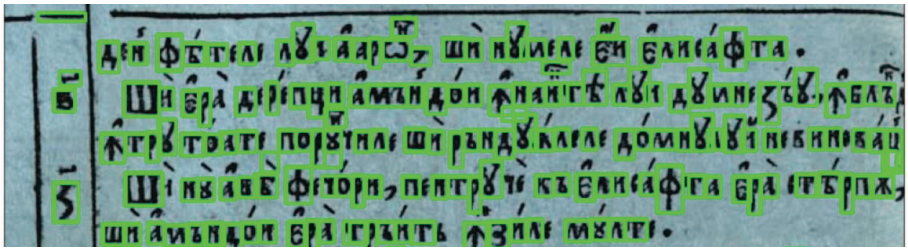

Figure 2. The contours of the characters in block of text from set A

The characters inside boxes (Figure 2) were cut and saved in two folders, for each class of fonts separately. From the first 10 pages of set A we obtained 17,155 characters, and from 9 pages of set B – 8,799 characters. Among the extracted characters, there are also elements of noise in the image – spots, accents without the basic character (letter), tilde, etc.

In the next subsection we try to remove unnecessary characters and keep only the letters. For this task we use *K-Means* clustering model implemented in *Scikit-learn* library (https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html).

3.3 Clustering the characters

We had to organize the characters extracted from text blocks in such a way that these letters become useful to train a neural network. So, we removed images that do not contain letters; grouped all the remaining images by clustering (this means that all the letters "ц" form a cluster, all the letters "з" form another cluster, and so on).

Before clustering, we had to make sure that all the images are of the same size. Each extracted character has the size of its boundary box, which varies widely. So, we resized all the images to the size of *50 by 50 pixels*.

K-Means clustering method was applied because it is simple and fast. The only thing we had to provide is the number of clusters. For a perfect result we have to end up with the exact number of letters in the Romanian Cyrillic alphabet – up to 47 letters considering the uppercase and lowercase. Each of them can appear as a lowercase or uppercase letter, which means that, in total, we expected 100 clusters (including punctuation marks).

After analyzing the clusters and deleting characters that are not letters and punctuation marks, we placed them in two bigger clusters: one – for set A and another cluster of characters – for set B. There are more than 10,000 characters in the set A, and set B contains 8,775 characters.

To form a well-organized dataset, we converted the dataset to *IDX format*. Since we train a neural network, we have to divide the images into two sets: a set of training and a set of testing data. For this purpose, we moved 1/3 of the former sets A and B to a test set. The output is 4 files: 2 files for image characters – for training and for testing, and another 2 files with the labels. The labels are the values *0* and *1*, where *1* is the class label for a character from set A, and *0* is the class label for a character from set B. After counting the examples in the dataset, we have: 21,212 *training examples* and 9,093 *test examples*. In Figure 3, we can see some examples from the training dataset.

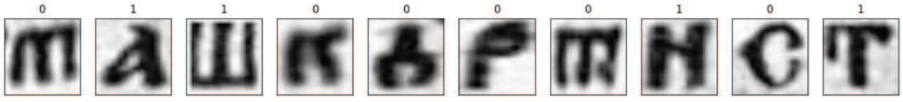

Figure 3. Samples from training dataset, where each image character has its class label – *0* or *1*

In the next section, we train a multilayer neural network and a convolutional neural network with the dataset prepared at this stage.

4 Neural network models

We trained two different neural network models to classify the characters in 2 classes of fonts. The first model is a multilayer perceptron (MLP), and the second one is a convolutional neural network.

The MLP model was implemented using Tensorflow and Keras (https://www.tensorflow.org/guide/keras). The input image matrix is reshaped into a vector of length x by y, where $x = 50$ and $y = 50$ – the dimensions of the image. Thus, we have an input layer of 2,500 neurons. We added a hidden layer with 128 neurons with the $ReLU$ activation function which is completely connected to the last layer. The last layer (output layer) contains a single neuron and a sigmoid function for its activation. At the training phase we set 300 epochs. The training phase lasted about 55 minutes without GPU.

The MLP model delivers an accuracy of 96.7%. Based on the confusion matrix (Figure 4), we computed the classification error – 3.3%.

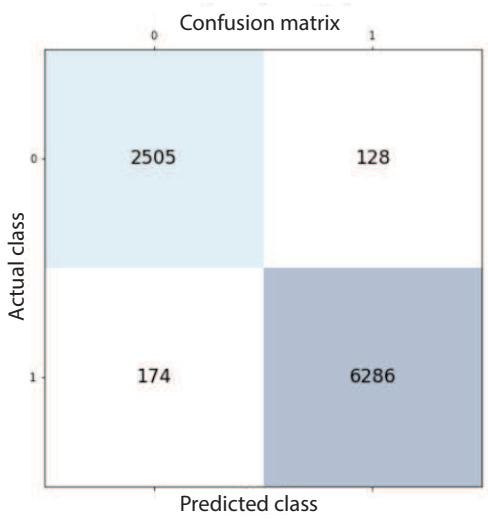

Figure 4. Confusion matrix of the MLP model based on the test dataset

The result we have seen after training the MLP model is pretty

good, but there is a chance that we can get a better accuracy using an architecture which is more specialized in image processing, namely the convolutional neural networks (CNN).

So, we have built a CNN model with 3 convolutional layers of 16, 32 and 64 neurons and one dense layer of 64 neurons. The output layer consists of 2 neurons as we use categorical cross-entropy as the loss function. The accuracy of our CNN model is 98.2% after training it through 100 epochs (see Figure 5). In this case, we used GPU power, and the training lasted about 11 minutes. We obtained a better result which can be the part of our final solution on classifying the font of a document X.

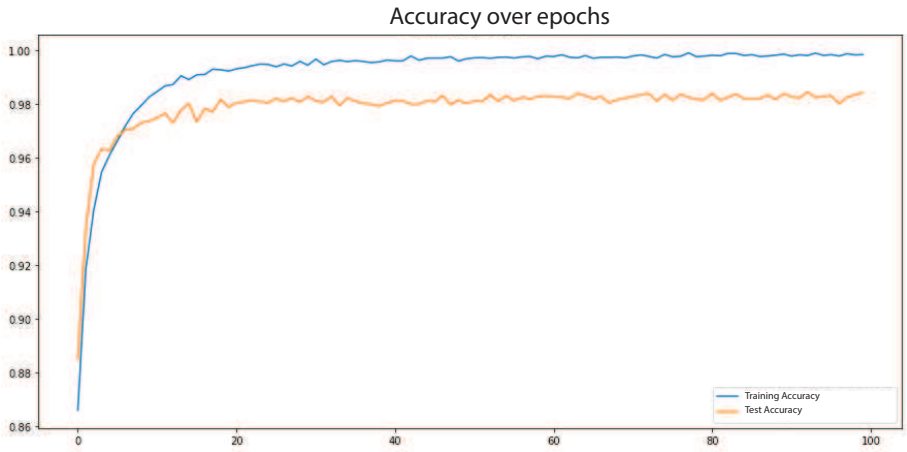

Figure 5. Accuracy chart for CNN model on training and test set

5 Conclusions

This paper is the extended and revised version of the conference paper [4] presented at WIIS 2021. In this paper we proposed a solution on how to classify the fonts in the Romanian Cyrillic documents of 17th century based on a mix of unsupervised and supervised machine learning technics. We used the unsupervised method K-Means to create the

dataset with the font characters and their classes and two different neural network architectures MLP and CNN to classify these characters. The obtained models perform well with the best accuracy of 98.2% by the CNN model. It can be deployed in our Model Selector tool [1] in order to identify the best OCR model to use on a particular Romanian Cyrillic document depending on its font style.

Acknowledgments. The research was supported by the project 20.80009.5007.22 "Intelligent information systems for solving ill-structured problems, processing knowledge, and big data".

References

[1] T. Bumbu, S. Cojocaru, A. Colesnicov, L. Malahov, and Ș. Ungur, "User Interface to Access Old Romanian Documents," in *Proceedings of the 4th Conference of Mathematical Society of Moldova CMSM4'2017, June 25-July 2, 2017*, pp. 479–482.

[2] R. Girshick, I. Radosavovic, G. Gkioxari, P. Doll ar, and K. He, "Detectron," 2018. Available: https://github.com/facebookresearch/detectron.

[3] C. Clausner, A. Antonacopoulos, and S. Pletschacher, "Prima Layout," in *Proceedings of 14th International Conference on Document Analysis and Recognition (ICDAR), 2017, volume 1*, pp. 1404–1410.

[4] Tudor Bumbu, "On Classification of 17th Century Fonts using Neural Networks," in *Workshop on Intelligent Information Systems (WIIS2021)*, (Chisinau, Republic of Moldova), October 14-15, 2021, pp. 58–64.

Tudor Bumbu

Received October 20, 2021

Vladimir Andrunachievici Institute of Mathematics and Computer Science
5, Academiei street, Chisinau, Republic of Moldova, MD 2028
E–mail: tudor.bumbu@math.md