FACTA UNIVERSITATIS
Series: **Automatic Control and Robotics** Vol. 10, N° 2, 2011, pp. 125 - 140

A COMPARISON BETWEEN TWO CHARACTER RECOGNITION APPROACHES[*]

UDC 004.032.26 004.93/'1 629.4.052.6

Lucian-Ovidiu Fedorovici

Dept. of Automation and Applied Informatics, "Politehnica" University of Timisoara,
Bd. V. Parvan 2, 300223, Timisoara, Romania,
E-mail: lucian.fedorovici@aut.upt.ro

Abstract. *This paper presents the architecture of an Optical Character Recognition (OCR) technology application based on two approaches, a multilayer neural network and a Support Vector Machine (SVM) classifier using Zernike moments for feature extraction. The performance comparison of the two approaches is based on the similar layout of most of the characters that must be recognized. The comparison shows that the improvement of the processing performance can be obtained by creating classes of blobs that use geometric similarities, and doing OCR only on the representative blob from each class.*

Key words: *OCR engine, character recognition, neural networks, SVM classifier, performance improvements*

1. Introduction

Optical Character Recognition (OCR) is the technology that transforms an image that contains text into an editable document in electronic format. The development of the technology started in the early 30's and has known an effervescent development in the last decade due to advances in computing power, image processing algorithms, feature extraction, classification algorithms and improvements in the printing quality and image retrieval.

An OCR engine contains several modules depending on the architecture, the character recognition being one the main module. Character recognition implies recognition of a single character or symbol, further on referred as blob, from a pre-segmented document. This module can be implemented using different approaches that can be grouped in two main classes: supervised classification and unsupervised classification.

Received November 27, 2011

[*] **Acknowledgement**: This work was supported by a grant of the Romanian National Authority for Scientific Research, CNCS – UEFISCDI, project number PN-II-ID-PCE-2011-3-0109. The author would like to thank Prof. Radu-Emil Precup for fruitful discussions regarding a systematic approach and student Andrei Tatar for implementation support assured.

The supervised classification approach is based on a classifier that is used to recognize various types of features that can be extracted from a blob image. These features can be grouped in: geometric moments [1-3], features extracted from blob contour [4] or descriptors [5, 6]. Zernike moments are used in image analysis and pattern recognition as invariant descriptors of the image shape. Studies proved that Zernike moments are superior to geometric moments in terms of feature representation capabilities and they are less sensitive to image noise [1, 7].

Artificial neural networks (ANN) can be used in OCR engines as simple classifiers or a combined feature extractor and classifier [8]-[10]. Although the accuracy obtained by a blob recognition module based on a neural network is high, the main disadvantage of a complex multilayer neural network is the computational overhead that it introduces in the OCR engine [10]. In the last ten years, a new classification algorithm was proposed by V. Vapnik, called the support vector machine (SVM) [11, 12]. The idea consists of mapping the space of the input examples into a high-dimensional (possibly infinite-dimensional) feature space. By choosing an adequate mapping, the input examples become linearly or almost linearly separable in the high-dimensional space [12]. SVM is usually applied to classification problems with a small number of categories, like classification of numeral strings [13]-[15], or font detection [16].

The paper presents the architecture of an OCR technology and focuses on comparing two algorithms used to recognize segmented blobs using two quality indicators, the time needed to recognize one blob and the percent of blob recognition accuracy. The first algorithm is based on a five layer convolutional neural network that works as both feature extractor and blob recognition, presented in Section 4.1 and the second algorithm is based on using an SVM classifier and Zernike moments for extracting blob features, presented in Section 4.2. In Section 3 is presented a Blob pre-Classifier that can be used to improve the time of the recognition process by grouping similar blobs into classes. A case study, results, comparisons between the two algorithms and discussions are offered in Section 5, and the conclusions are drawn in Section 6.

2. Architecture of the OCR Engine

The architecture of the OCR engine is depicted graphically in Fig. 1 [10]. The OCR engine proposed in this paper was designed for recognizing printed text with medium or low quality.

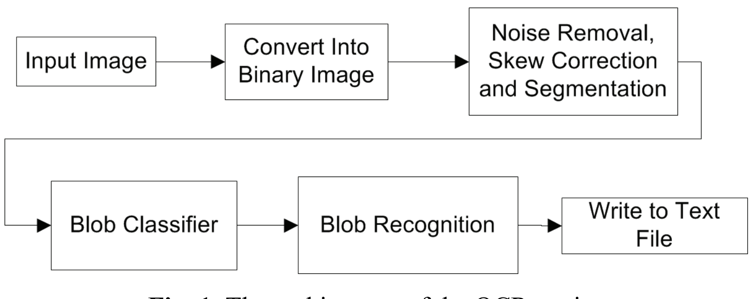
Input Image
Convert Into Binary Image
Noise Removal, Skew Correction and Segmentation
Blob Classifier
Blob Recognition
Write to Text File

Fig. 1. The architecture of the OCR engine

The input of the OCR engine represents the image of a scanned document that was previously printed. The first module is responsible for the conversion of the image into a binary representation that will be further used by the OCR engine. The Noise Removal, Skew Correction and Segmentation module takes the binary image and performs several operations, like: background removal, skew correction, text detection and text segmentation. The output of this module represents a set of blobs and their absolute position on the image.

The Blob Classifier is a module used to pre-classify the set of blobs produced by the previous module based on geometric similarities. The output of the classifier is a collection of blob classes. The first blob from each class can then be presented to the Recognition module and thus, improving the overall time that the OCR engine requires to process an image.

The paper focuses on the Recognition module and presents two different approaches and the advantages and disadvantages of each: a convolution neural network and a SVM classifier that uses Zernike moments as blob features.

The output of the OCR engine represents the text file that results after the recognition of all blobs.

3. Blob Classifier Module

As mentioned in the previous paragraph the role of the Blob Classifier component is to group the similar blobs into classes. Fig. 2 depicts graphically the structure of the component. The input of the classifier is the collection of blobs that need to be grouped and three thresholds parameters that are used to determine if a blob belongs to a class or not. The output of the component is a collection of blob classes. The first element of each class represents the element that defines that class.

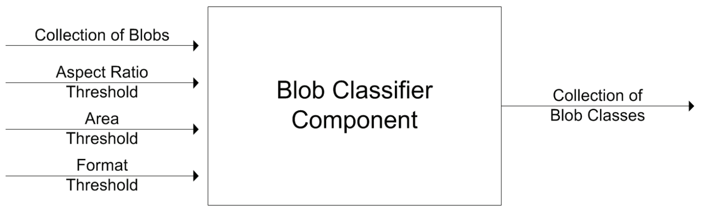

Fig. 2. The Blob Classifier Component

The three configurable thresholds are the aspect ratio threshold (A_r), the area threshold (A) and the format threshold (F). The Blob Classifier, first presented in [10] was updated to support parallel processing to further improve the time needed by the OCR engine to process a document. The thresholds are calculated as follows:

$$A_r = \frac{|a_i - a_j|}{\max(a_i, a_j)}, \quad a_k = \begin{cases} -(w_{b_k} / h_{b_k}), w_{b_k} > h_{b_k} \\ (h_{b_k} / w_{b_k}), w_{b_k} \le h_{b_k} \end{cases} \tag{1}$$

$$A = \frac{|A_i - A_j|}{\max(A_i, A_j)}, \quad A_k = \sum_{x,y} f_k(x, y) \tag{2}$$

$$F = \frac{\sum_{x,y} (f_i(x,y) - f_j(x,y))}{2wh} \tag{3}$$

where w_{b_k} and h_{b_k} represent the width and height of the segmented blob b_k image, and $f_k(x,y)$ represents the value of the pixel with the coordinates (x,y) that correspond to the blob b_k.

The operations performed by the Blob Classifier are simple pixel by pixel comparisons and are performed during a single walk through the two blob images. If any of the results exceeds the specified threshold (A_r, A or F), the classifier passes to the next class or creates a new one if all classes were verified.

The blob classes are created dynamically based on the collection of input blobs extracted from the processed image. That means that when the classification process is started there are no predefined blob classes. The number of classes that are generated by the classifier component is strictly related to the specified threshold parameters and the variation of the type of font used to edit the document [17]. If the threshold parameters are set to zero, the classifier will create a new class for each blob from the input collection. If the thresholds are set too high, the classifier will group blobs that are different in the same class.

4. Blob Recognition Modules

4.1. Blob recognition module with convolutional neural network

Studies have shown that convolutional neural networks have been found well suited for visual document analysis tasks and character recognition [18]. This type of neural networks was designed specifically to recognize two-dimensional shapes with a high degree of invariance to translation, scaling, skewing and other forms of distortions. The overall architecture of the convolutional neural network that was used for the Recognition module is depicted graphically in Fig. 3 [8].

The general strategy of a convolutional network is to extract features, like orientated edges, corners or end-points. Each neuron takes its synaptic inputs from a local receptive field, called a convolutional kernel from the previous layer, thereby forcing it to extract local features. Once a feature has been extracted, its location becomes less important as long as its position relative to other features is approximately preserved. First two computational layers of the networks are composed of multiple feature maps. Each feature map forms a plane within which individual neurons are constrained to share the same set of synaptic weights. This form of structural constraint has the following beneficial effects: shift invariance and reduction in the number of free parameters through weight sharing. The third and the forth layer, the output layer, forms a trainable universal classifier [19].

The input layer, made up of 29×29 neurons, receives the binary image of different blobs that have been previously centered and normalized in size. The size of the input layer was determined by the size of the convolutional kernel which was chosen to be 5×5 pixels taking in consideration the following constraints: the width of the kernel must be centered on a unit – odd size; it must have sufficient overlap so as not to lose information and yet not to have redundant computation – three would have been too small with only one unit overlap and seven would have been too large with over 70% overlap. With no image padding, a sub-sampling of two and a kernel of size five the nearest value which generated an integer after two layers of convolution was 29×29 [9].

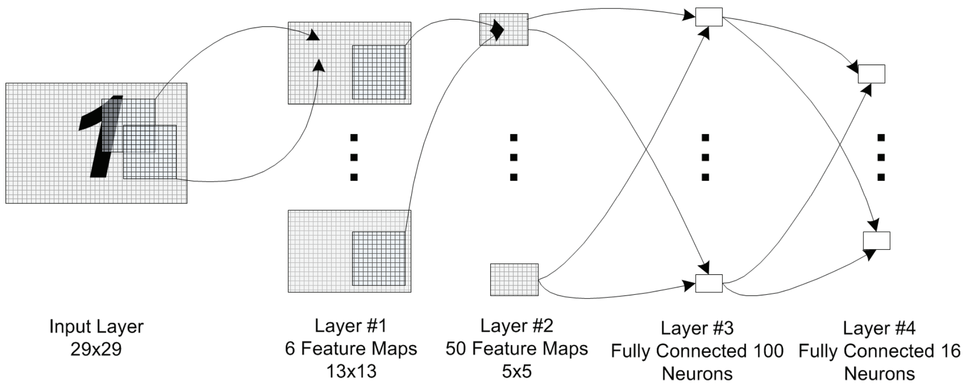

Fig. 3. The architecture of the convolutional neural network

The first hidden layer is a convolutional layer with six feature maps, each feature map consisting of 13x13 neurons. Each neuron in each feature map is a 5x5 convolutional kernel of the input layer. The convolutional kernel is moved by two positions for each neuron from the feature map resulting in 13 positions where the kernel will fit in each row and 13 positions where the kernel will fit in each column. The connections are depicted graphically in Fig. 4. If the following notations are considered:

- $x^{i,j}_{2,f_k}$ represents the output of the neuron from the second layer in the f_k feature map at coordinates (i, j), where $k = \overline{1..6}$ and $i, j = \overline{1..13}$,
- $x^{i,j}_{1}$ represents the output of the neuron from the first layer at coordinates (i,j), where $i, j = \overline{1..29}$,
- w^{l}_{2,f_k} represents the weight corresponding to the f_k feature map for the l connection from the second layer,
- $F()$ represents the activation function,

then (4) represents the mathematical model that describes the connection of the first hidden layer to the input one:

$$x^{i,j}_{2,f_k} = F\left(\sum_{l=1}^{25} (w^{l}_{2,f_k} x_1^{2(i-1)+l \bmod 5, 2(j-1)+l \bmod 5}) + w^{26}_{2,f_k}\right) \quad (4)$$

The first convolutional layer will have:

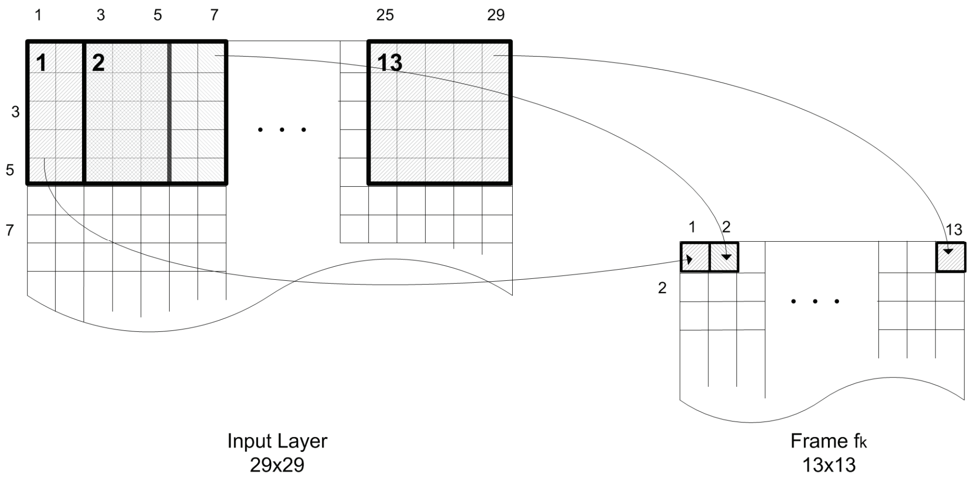

Fig. 4. Connections from the input layer to the frame f_k from the first hidden layer

$$\begin{cases} 13\times 13\times 6 = 1014 & \textit{neurons} \\ (5\times 5+1)\times 6 = 156 & \textit{weights} \\ (25+1)\times 1014 = 26364 & \textit{connections} \end{cases} \tag{5}$$

The second hidden layer is also a convolutional layer, but with 50 feature maps, each feature map consisting of 5×5 neurons. Each neuron in each feature map is a 5×5 convolutional kernel of corresponding areas of all six feature maps of the previous layer. Considering the notations introduced before and:

- $x_{3,f_k}^{i,j}$ represents the output of the neuron from the third layer in the f_k feature map at coordinates (i,j), where $k=\overline{1..50}$ and $i,j=\overline{1..5}$,
- w_{3,f_k}^{l} represents the weight corresponding to the f_k feature map for the l connection from the third layer,

the mathematical model of the second hidden layer is:

$$x_{3,f_k}^{i,j} = F\left(\sum_{n=1}^{6}\left(\sum_{l=1}^{25}(w_{3,f_k}^{26(n-1)+l} \times x_{2,f_n}^{2(i-1)+l \bmod 5,(j-1)+l \bmod 5}) + w_{3,f_k}^{26n}\right)\right) \tag{6}$$

The second convolutional layer will have:

$$\begin{cases} 5\times 5\times 50 = 1250 & \textit{neurons} \\ (5\times 5+1)\times 6\times 5 = 7800 & \textit{weights} \\ (25+1)\times 1250 = 32500 & \textit{connections} \end{cases} \tag{7}$$

The third hidden layer is a fully connected layer consisting of 100 neurons. By varying the number of neurons from this layer we can control the interpolation capability of the classifier. The network used in this paper had 100 neurons. There are therefore $(1250+1)\times 100 = 125100$ connection and weights, where +1 comes from the bias.

The last layer of the neural network, the output layer, represents the ASCII code of the blob that was presented at the input. Therefore, there are 16 neurons on the last layer, $(100+1)\times 16 = 1616$ weights and connections, where +1 comes from the bias. Using the following notations:

- x_n^i represents the output of the ith neuron in layer n,
- $w_n^{i,j}$ represents the weight that the ith neuron in layer n applies to the output of the jth neuron from layer *n-1*,
- C_n is the number of neurons from layer *n-1*,

the general mathematical model for a fully connected layer is given in (8):

$$x_n^i = F\left(\sum_{l=1}^{C_n} w_n^{i,l} x_{n-1}^l\right) \tag{8}$$

The activation function *F*() was selected to be a scaled version of the hyperbolic tangent because it is symmetric and the derivate is very easy to obtain using only the output of the function, as expressed in (9) and (10):

$$x = F(y) = \tanh(y) = \frac{\sinh(y)}{\cosh(y)} \tag{9}$$

$$\frac{dF}{dy} = \frac{d}{dy}\left(\frac{\sinh(y)}{\cosh(y)}\right) = \frac{\cosh^2(y) - \sinh^2(y)}{\cosh^2(y)} = 1 - \tanh^2(y) = 1 - x^2 \tag{10}$$

The function was scaled to vary in the ±1.7 interval to allow the neural network to be trained in the ± 1 interval and avoid saturating the output units [9].

The neural network was trained using the backpropagation algorithm and an input set of about 20000 blobs. The input set contained digits, upper case and lower case letters and the following special characters: ampersand, dollar, hyphen, left and right round parenthesis, percent, comma and full stop. To improve the neural network's interpolation capability, the following techniques were used in the training process: the training set was passed through the neural network in a randomized order in each epoch and distortions were applied to the characters [9]. First technique forces the network to not settle on weights that emphasize attributes from a single class of characters and causes the weights to change for each input. The second technique was used to extend the training set by applying the following distortions to the input characters: elastic, scale and rotation. Each type of distortion had a severity factor that was determined experimental. For printed characters the distortions helped simulating the loose in quality for poor printed documents combined with text printed using small sizes.

The initial values for the weights were randomly selected to be in the ±0.05 interval. The training process started with a learning rate of 0.0005. The learning rate was adjusted after five epochs or if the difference between the error of the current epoch and the previous one was less than the value of the learning rate. The error for one epoch was computed as being the sum of the mean square errors computed for each blob. The learning rate was adjusted by multiplying it with 0.990.

To improve the performance of the OCR module, the neural network was designed to support parallel processing. The number of threads is determined dynamically based on the number of cores available on the machine on which the processing is made. The multi-threading technique was tried also in the training process but the results were not positive. Moreover the multi-threading technique caused the weights of the network to change without converging to a minimum.

4.2. Blob recognition module with SVM classifier

The SVM is a supervised learning algorithm, useful for recognizing subtle patterns in a complex dataset. The SVM used in the blob recognition component is an extension of the original SVM that allows the classification to be performed on more than two classes. This type of SVM is called C-SVC (C-Support Vector Classification, Binary Case).

The mathematical relations that describe the operation of an SVM C-SVC classifier are described as follows. Given a training set of instance-label pairs: (x_i, y_i) where $i = \overline{1..I}$, $x_i \in R^n$ and $y \in \{-1,+1\}^I$ the SVM gives the solution to the following optimization problem:

$$\min_{\omega,b,\xi} \frac{1}{2}\omega^T\omega + C\sum_{i=1}^{I}\xi_i \; , \tag{11}$$

subject to:

$$y_i(\omega^T\phi(x_i)+b) \geq 1-\xi_i \, , \; \xi_i \geq 0 \tag{12}$$

SVM finds a linear separation hyperplane with the maximal margin in this higher dimensional space. In (11), $C > 0$ is the penalty parameter of the error term [12]. The decision function is:

$$\operatorname{sgn}\left(\sum_{i=1}^{I} y_i\alpha_i K(x_i, x) + b\right) \tag{13}$$

The kernel of the SVM is a Gaussian radial basis function (RBF) that will determine an infinite Hilbert space for the corresponding feature space. The expression of the kernel function is:

$$K(x_i, x_j) = \exp\left(-\frac{\left\|x_i - x_j\right\|^2}{2\sigma^2}\right) \tag{14}$$

The blob recognition module uses an implementation of the SVM classifier in C# [20]. To train the SVM, the cross-validation and grid-search methods were used [21].

The Zernike moments are known for their high accuracy in image reconstruction and feature extraction [22]. Complex Zernike moments are constructed using a set of complex polynomials defined over the polar coordinates inside a unit circle, $(x^2+y^2) \leq 1$. The two-dimensional Zernike moments of order p with repetition q of a function $f(x,y)$ that is subject to description are:

$$Z_{pq} = \frac{p+1}{\pi}\int_x\int_y f(x,y)[V_{pq}(x,y)]^* dxdy \tag{15}$$

where:

- $p = 0,1,2,...,\infty$ – defines the order,
- $|q| \le p$ and $p - |q| = even$,
- * denotes the complex conjugate of $V_{pq}(x, y)$.

The Zernike polynomial expressed in polar coordinates is:

$$V_{pq}(r, \theta) = R_{pq}(r) e^{jq\theta} \tag{16}$$

where:

- (r, θ) are defined over the unit disc,
- $j = \sqrt{-1}$,
- $R_{pq}(r)$ is the orthogonal radial polynomial:

$$R_{pq}(r) = \sum_{s=0}^{\frac{p-|q|}{2}} (-1)^s \frac{(p-s)!}{s!\left(\frac{p+|q|}{2} - s\right)!\left(\frac{p-|q|}{2} - s\right)!} \cdot r^{p-2s} \tag{17}$$

For a discrete image, the Zernike polynomial for a pixel located at *(x, y)* coordinates is calculated in terms of:

$$Z_{pq} = \frac{p+1}{\pi} \sum_x \sum_y f(x, y) [V_{pq}(x, y)]^* \tag{18}$$

To calculate the Zernike moments, the image is first mapped to the unit disc using polar coordinates, where the center of the image is the origin of the unit disc. The mapping from Cartesian to polar coordinates is:

$$\begin{cases} x = r\cos\theta \\ y = r\sin\theta \end{cases} \tag{19}$$

$$r = \sqrt{x^2 + y^2}, \theta = \tan^{-1}\left(\frac{y}{x}\right), \ -1 < x, y < 1 \tag{20}$$

The blob recognition module uses Zernike moments of order ten (p = 10). Based on (15), the total number of moments that can be calculated for each pixel from an image for order 10 is 36. To improve the time that the component needs to process a blob, the Zernike polynomial coefficients are pre-calculated. The images processed by the component need to be normalized to a size of 32×32 pixels.

To process a blob, the recognition module first needs to be initialized. The initialization process consists of loading the SVM data and the pre-calculated Zernike coefficients of order ten for an image of 32×32 pixels that is mapped inside the unit circle using (18)–(20). The blob processing is performed through the following steps, after the module was initialized:

1. The module checks if the image has the correct size and if not it will scale it to the 32×32 pixels using a bicubic resampling algorithm.
2. To apply the Zernike moments to each pixel from the image, the polynomial (18) is split into real and imaginary parts:

$$\begin{cases} V_{pq}^{\mathrm{Re}}(r,\theta) = R_{pq}(r)\cos(q\theta) \\ V_{pq}^{\mathrm{Im}}(r,\theta) = R_{pq}(r)\sin(q\theta) \end{cases} \tag{21}$$

   For each pixel, the module will compute 36 complex numbers that are represented in memory as real and imaginary parts.
3. The next step consists of calculating the gravity center of the image using (22).

$$G_x = \sum_x (I \times x),\ G_y = \sum_y (I \times y)$$
$$I = \begin{cases} 0 & f(x,y) < 50 \\ 1 & f(x,y) \geq 50 \end{cases} \tag{22}$$

   If the value of the pixel is smaller than the predefined threshold, 50, the pixel will not be used to compute the center of gravity of the image.
4. Compute the aspect ratio of the image A_r :

$$A_r = \frac{w}{h} \tag{23}$$

   where w represents the width of the image and h represents the height.
5. Create the array of features that will be used by the SVM classifier. The array of features includes the Zernike moments that were calculated at step 2, the gravity center of the image from step 3, the aspect ratio and the inverse of the aspect ratio from step 4 and the height and width of the image.
6. Scale the array of features in the ±1 interval. For the recognition process, the scaled array of features will be used.

The SVM based Recognition module is composed of a primary classifier and seven sub-classifiers. The primary classifier was trained to recognize blobs or group of blobs if the blobs have similar features. The groups of blobs recognized by the primary classifier are: '0o', '1iltIjf', 'B83aes5', '9g', 'hb6', 'vy', '2z'. The rest of the characters that are not defined in a group of characters are recognized directly by the primary classifier. For each group of blobs, a sub-classifier was trained to recognize only those types of blobs. The recognition process is performed in two steps: first the primary classifier is used to recognize the blob. If the blob belongs to one of the groups, then the sub-classifier that corresponds to that group is used to recognize the exact character that the blog represents.

5. Case Study - Results

The case study proposed is a comparison between the two blob recognition algorithms using two quality indicators, mainly the time needed to recognize a blob, (24), and the blob recognition accuracy, (25):

$$T_b = \frac{T_t}{N} \tag{24}$$

$$A_b = \frac{N_c}{N} 100 [\%] \tag{25}$$

where T_t is the total time needed to recognize N blobs and N_c is the number of blobs that were correctly recognized.

Both recognition modules were trained using the same set of approximately 20,000 blobs, varying from digits to lower and upper case characters and some special characters, like ampersand, dollar, hyphen, percent, comma, left and right round parenthesis. For both recognition modules, the image of the blob needs to be normalized in advanced, to a size of 29×29 pixels for the neuronal network based recognition module, further referred as NN Recognition module, and to a 32×32 pixels for the SVM based classifier using Zernike moments as features, further referred as SVM Recognition module.

The case study was performed using a testing set of 10,000 characters that were extracted and preprocessed from conventional printed documents. The documents were scanned at 300 DPIs as black and white images. The documents that were used to create the test set were selected so that they would contain both fixed size font types and variable size font types and also contained text edited with both small and large fonts, varying from 8 pixels to 40 pixels.

The case study follows five scenarios: the first one does not use the Blob Classifier module and the results are obtained using only the blob recognition module. The next four scenarios use the Blob Classifier module and the blob recognition module is used for identifying only the first blob from each class. The results are labeled in Fig. 5, 6, 7, 8 with: (a) only the Blob Recognition module is used; (b) Blob Classifier is used with the following parameters: $A_r = 0.1$, $A = 0.1$ and $F = 0.05$, (c) parameters of the Blob Classifier are: $A_r = 0.1$, $A = 0.1$ and $F = 0.055$; (d) the parameters of the Blob Classifier are: $A_r = 0.1$, $A = 0.1$ and $F = 0.06$; (e) the parameters of the Blob Classifier are: $A_r = 0.08$, $A = 0.08$ and $F = 0.08$. The parameters of the Blob Classifier were obtained empirically, starting from initial values of 0.1 for all parameters and fine tuning them based on the results obtained, as presented in [10].

Both blob recognition modules can use the parallel processing capability of current computers. The time needed to recognize a blob was measured for each scenario from the case study using one, two and four processing threads. In Fig. 5 and Fig. 7 for each case study, the T_b has three different values, where the highest one was obtained using only one processing thread and the lowest one was obtained using four processing threads.

The first results that are presented are the T_b, in Fig. 5 and A_b, in Fig. 6, obtained using NN Recognition module. The T_b is slightly improved from the one presented in [10] due to the fact that the Blob Classification module was modified to support parallel processing. This capability improved the time to recognize one blob with approximately 10%.

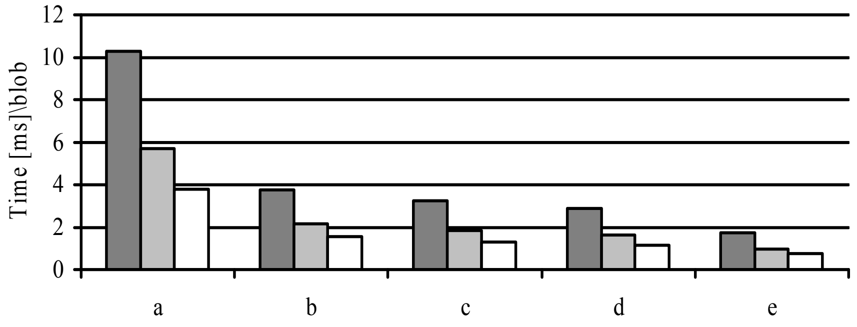

Fig. 5. T_b using NN Recognition module

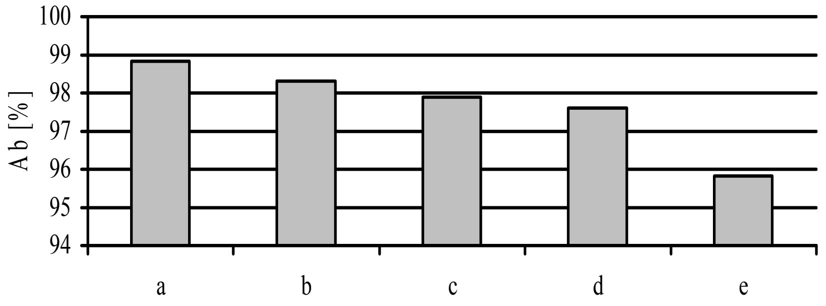

Fig. 6. A_b using NN Recognition module

By comparing scenario (a) and (b), it is clear that the time to recognize one blob can be improved by at least a half only by using the Blob Classifier module without losing the accuracy of the recognition process. Because of the simplicity of the classifier, some very similar blobs will be misclassified if the values of the thresholds are decreased causing the loss of accuracy, as best seen in scenario (e), even if the value T_b is improved.

The results obtained by the SVM Recognition module are presented in Fig. 7 and Fig. 8. In this scenario the value of T_b is 60% lower than the one obtained by the NN Recognition module. Using the Blob Classifier in this case will not improve dramatically the performance from the time point of view due to the fact that the time that the Blob Classifier needs to create the blob classes is approximately equal to the time needed by the SVM Recognition module to recognize the blobs.

The value of A_b obtained using the SVM Recognition module is 3.13% less than the one obtained with the NN Recognition module. It is interesting to notice that in some cases the Blob Classifier can even improve the accuracy percentage obtained by the SVM. This improvement is related to similar blobs that the SVM does not recognized correctly individually but can be pre-classified using simple comparisons.

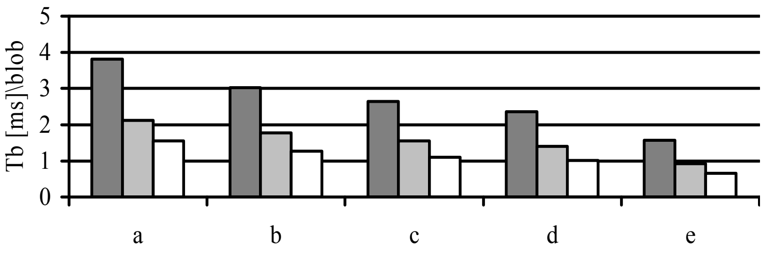

Fig. 7. T_b using SVM Recognition module

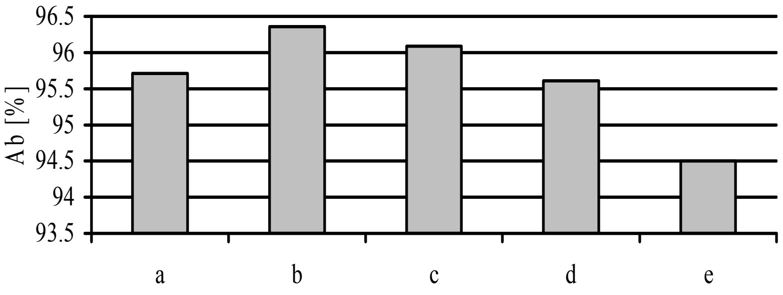

Fig. 8. A_b using SVM Recognition module

These results show that the interpolation capability of the NN Recognition module is better than the one of the SVM Recognition module. Even if the accuracy of the NN Recognition module is better, the time that it needs to recognize a single blob, without using the Blob Classifier first is three times greater. Another problem of the NN Recognition module is the process of training the neural network due to the fact that the weights can converge to a local minimum. To overcome this problem, distortions were applied to the training set and the data was presented to the network in randomized order. Due to the large set of training and techniques applied in conjunction with the backpropagation algorithm, the converging of the value of the weights is very slow and requires a large computational power.

The SVM Recognition module was trained in less than half an hour using the same set of training data. This is a major difference that could make the SVM Recognition module a better solution for the OCR engine for multiple reasons: the training set can be further extended without the need of spending a lot of time fine tuning the recognition module; if the segmentation module changes, a new set of data can be created using the new module that can be used to re-train the recognition module. It is very important that the recognition module is trained using a set of blobs that is created with the segmentation module due to the fact that different segmentation algorithms can produce different outputs which will have a negative impact on the accuracy of the OCR engine. Another advantage of the SVM Recognition module is that it can be extended to use other blob features, like Fourier Descriptors, or other invariant moments to recognize a blob, while the structure of the convolutional neural network is rigid and does not allow any extending.

6. CONCLUSIONS

What has been presented in this paper is the architecture of an OCR engine extended to contain a blob pre-classifier that can be used to improve the overall performance by creating classes of blobs using geometrical similarities and allow the recognition module to recognize only the representative blob from each class. The paper proposes two algorithms for blob recognition, one based on a five layer convolutional neural networks that acts as both feature extractor and recognizer and the second one based on a SVM classifier that uses Zernike moments as blob features.

Both recognition modules were trained and tested with the same sets of data extracted from real documents. It has been shown that the convolutional neuronal network has a higher accuracy percentage but requires more computation power and time. Another disadvantage of the neural network is represented by the training process which is more difficult and requires additional techniques to avoid converging to a local minimum that will lead to poor results when presented with real data.

Even though the accuracy was less than the one obtained by the convolutional neural network, the SVM classifier has several advantages: it requires less time to train and the time needed to recognize a blob is almost three times smaller. Another advantage of the SVM classifier is that it allows a degree of flexibility in the architecture of the recognition module by extending the features used to recognize a blob. Besides the Zernike moments, the SVM classifier can be easily extended to use other features, like Fourier descriptors, invariant moments or the contour of the blob.

The future research will concern the reduction of the training, fine tuning of the neural network and extending the features used in conjunction with the SVM classifier. The combination with several optimization algorithms and modeling approaches [23]-[35] will be taken into consideration.

REFERENCES

1. S.O. Belkasim, M. Shridhar, M. Ahmadi, "Pattern recognition with moment invariants: A comparative study and new results", Pattern Recognition, vol. 14, pp. 1117–1138, Dec. 1991.
2. J. Flusser, T. Suk, "Pattern recognition by affine moment invariants", Pattern Recognition, vol. 26, pp. 167–174, Jan. 1993.
3. J. Flusser, T. Suk, "Affine moment invariants: a new tool for character recognition", Pattern Recognition Letters, vol. 15, pp. 433–436, Apr. 1994.
4. T. Zhang, X. Wang, C. Chen, J. Liu, "Connected Numeral Strings Segmentation Based on the Combination of Characteristic Position and Contour Detecting", Proc. ICDIP, pp. 81–84, 2009.
5. D. Zhang and G. Lu, "A comparative study of three region shape descriptors", Proc. DICTA, pp. 21–22, 2002.
6. K. Arbeter, W. E. Snyder, H. Burkhardt, G. Hirzinger, "Application of affine-invariant Fourier descriptors to recognition of 3-D objects," IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 12, pp. 640–647, July 1990.
7. C.W. Chong, R. Mukundan and P. Raveendran, "An efficient algorithm for fast computation of pseudo-Zernike moment", IJPRAI, vol. 17, pp. 1011–1023, 2003.
8. Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition", Proc. IEEE, vol. 86, pp. 2278–2324, 1998.
9. P.Y. Simard, D. Steinkraus and J. Platt, "Best practices for convolutional neural networks applied to visual document analysis", ICDAR, pp. 958–962, 2003.
10. L.-O. Fedorovici, E. Voisan, F. Dragan, D. Iercan, "Improved neural network OCR based on preprocessed blob classes", Proc. ICCC-CONTI 2010, Timisoara, Romania, pp. 559–564, 2010.

11. V.N. Vapnik, "Statistical Learning Theory", John Wiley and Sons, New York, USA, 1998.
12. V.N. Vapnik, "An overview of statistical learning theory", IEEE Trans. Neural Netw., vol. 10, pp. 988–999, Sep. 1999.
13. A. Bellili, M. Gilloux, P. Gallinari, "An hybrid MLP-SVM handwritten digit recognizer", Document Analysis and Recognition, pp. 28–32, 2001.
14. S.A. Mahmoud, S.O. Olatunji, "Automatic recognition of off-line handwritten Arabic (Indian) numerals using support vector and extreme learning machines", Int. Journal of Imaging and Robotics, vol. 2, pp. 34–53, 2009.
15. J. Dong, A. Krzyzak, C. Suen, "An improved handwritten Chinese character recognition system using support vector machine", Pattern Recognition Letters, vol. 26, pp. 1849–1856, 2005.
16. R. Ramanathan, L. Thaneshwaran, V. Viknesh, T. Arunkumar, P. Yuvaraj, K.P. Soman, "A Novel Technique for English Font Recognition Using Support Vector Machines", ARTCom, pp. 766–769, 2009.
17. L.-O. Fedorovici, F. Dragan, "A comparison between a neural network and a SVM and Zernike moments based blob recognition modules", Proc. SACI 2011, Timisoara, Romania, pp. 253–258, 2011.
18. Y. Tay, P. Lallican, M. Khalid, C. Viard-Gaudin, S. Knerr, "An offline cursive handwriting word recognition system", Proc. IEEE Region 10 TENCON Conf., pp. 519–524, 2001.
19. S. Haykin, Neural networks. A comprehensive foundation, 2nd ed., Prentice Hall International, 1999.
20. M. Johnson, "SVM.NET" [Online]. Available: http://www.matthewajohnson.org/software/svm.html 2009.
21. C. Hsu, C. Chang, C.-J. Lin, "A Practical Guide to Support Vector Classification", [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/, 2008.
22. A. Khotanzad, Y.H. Hong, "Invariant image recognition by Zernike moments", IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 12, pp. 489–197, May 1990.
23. P. Baranyi, K.F. Le, Y. Yam, "Complexity reduction of singleton based neuro-fuzzy algorithm", Proc. IEEE Intl. Conf. Systems, Man, and Cybernetics (SMC'00), Nashville, TN, USA, vol. 4, pp. 2503–2508, 2000.
24. P. Baranyi, D. Tikk, Y. Yam, R.J. Patton, "From differential equations to PDC controller design via numerical transformation", Comp. Ind., vol. 51, pp. 281–297, Aug. 2003.
25. S. Blažič, I. Škrjanc, D. Matko, "Globally stable direct fuzzy model reference adaptive control", Fuzzy Sets Syst., vol. 139, pp. 3–33, Oct. 2003.
26. I. Škrjanc, S. Blažič, S. Oblak, J. Richalet, "An approach to predictive control of multivariable time-delayed plant: Stability and design issues", ISA Trans., vol. 43, pp. 585–595, Oct. 2004.
27. J. Vaščák, "Evolutionary migration algorithms for scheduling", Proc. 3rd Slovakian-Hungarian Joint Symposium on Applied Machine Intelligence (SAMI 2005), Herľany, Slovakia, pp. 21–32, 2005.
28. Z. C. Johanyák, S. Kovács, "Fuzzy rule interpolation based on polar cuts", in Computational Intelligence, Theory and Applications, B. Reusch, Ed. Berlin, Heidelberg, New York: Springer-Verlag, pp. 499–511, 2006.
29. D. Hladek, J. Vaščák, P. Sinčák, "Multi-robot control system for pursuit-evasion problem", J. Electr. Eng., vol. 60, pp. 143–148, Jun. 2009.
30. Z. C. Johanyák, "Student evaluation based on fuzzy rule interpolation", Int. J. Artif. Intell., vol. 5, pp. 37–55, Sep. 2010.
31. J. A. Iglesias, P. Angelov, A. Ledezma, A. Sanchis, "Evolving classification of agents' behaviors: a general approach", Evolving Syst., vol. 1, pp. 161–171, Oct. 2010.
32. Z. Jovanović, D. Antić, Z. Stajić, M. Milošević, S. Nikolić, S. Perić, "Genetic algorithms applied in parameters determination of the 3D crane model", Facta Universitatis Series Automatic Control and Robotics, vol. 10, no. 1, pp. 19–27, Jan. 2011.
33. A. Garcia, A. Luviano-Juarez, I. Chairez, A. Poznyak, T. Poznyak, "Projectional dynamic neural network identifier for chaotic systems: Application to Chua's circuit", Int. J. Artif. Intell., vol. 6, pp. 1–18, Mar. 2011.
34. R.-E. Precup, R.-C. David, E. M. Petriu, S. Preitl, M.-B. Rădac, "Fuzzy control systems with reduced parametric sensitivity based on simulated annealing", IEEE Trans. Ind. Electron., DOI: 10.1109/TIE.2011.2130493.
35. M.-B. Rădac, R.-E. Precup, E. M. Petriu, S. Preitl, "Application of IFT and SPSA to servo system control", IEEE Trans. Neural Netw., DOI: 10.1109/TNN.2011.2173804.

POREĐENJE IZMEĐU DVA PRISTUPA PREPOZNAVANJA KARAKTERA

Lucian-Ovidiu Fedorovici

Ovaj rad predstavlja arhitekturu tehnologije Optičkog Prepoznavanja Karaktera (OCR) zasnovane na dva pristupa, višeslojne nauronske mreže i Support Vector Machine (SVM) klasifikatora koji koristi Zernikove momente za izdvajanje karakteristika. Poređenje performansi dva pristupa se bazira na sličnom rasporedu većine karaktera koji treba da budu prepoznati. Poređenje pokazuje da se poboljšanje procesnih performansi može postići stvaranjem klasa bitova binarne slike koji koriste geometrijske sličnosti, i obavljanjem OCR-a samo na reprezentativnim bitovima u svakoj klasi.

Ključne reči: *OCR mehanizam, prepoznavanje karaktera, neuralne mreže, SVM klasifikator, poboljšanje performansi*