A SVM Based Off-Line Handwritten Digit Recognizer

Renata F. P. Neves, Alberto N. G. Lopes Filho, Carlos A.B.Mello, *IEEE Member*, Cleber Zanchettin
Center of Informatics, Federal University of Pernambuco
CIn - UFPE
Recife, Brazil
{rfpn, anglf, cabm, cz}@cin.ufpe.br

Abstract— **This paper presents an efficient method for handwritten digit recognition. The proposed method makes use of Support Vector Machines (SVM), benefitting from its generalization power. The method presents improved recognition rates when compared to Multi-Layer Perceptron (MLP) classifiers, other SVM classifiers and hybrid classifiers. Experiments and comparisons were done using a digit set extracted from the NIST SD19 digit database. The proposed SVM method achieved higher recognition rates and it outperformed other methods. It is also shown that although using solely SVMs for the task, the new method does not suffer when considering processing time.**

Keywords-OCR; MLP;SVM; Handwritten Digit Recognizer

I. INTRODUCTION

This paper presents a technique developed for recognition images of handwritten digits. This is a task of paramount importance for businesses and enterprises of most diverse areas. A practical example would be automatic bank check processing. This type of business requires a solution that provides high successful rates in handwritten digit recognition in order to correctly process the values of the checks. The need for a method with low error rates is clear, since an error in interpreting the value of a check can cost to the client or to the bank high amount of money. Handwritten character recognition is a task which presents an elevated level of difficulty due to the high variety of different writings. The constant quest for classifiers with smaller error rates is justified by the high costs associated with misclassifying a character.

A common method for constructing a classifier involves the use of a Multi-Layer Perceptron (MLP) [1][2]. Other techniques have been developed using the MLP structure together with different methods. Bellili et al. [3], for example, proposed a hybrid method in order to boost the MLP performance; their method coupled MLP and Support Vector Machine (SVM) [4]. Other authors [1][4][5][6] also use MLP or SVM, but with variations on the way they are employed or the problem to which they are applied.

The rest of this manuscript is organized as follows: Section 2 discusses related works in the area of character recognition; in Section 3, the proposed method is explained; Section 4 shows the results of the new approach compared to existing methods, and Section 5 concludes the paper.

II. RELATED WORKS

This section offers a brief overview of some techniques used in automatic handwritten character recognition. Some methods can be considered as classical, while others may employ subtle variations or combination of other known methods. The objective of this section is to place our method in context with the existing techniques, as well as explain the methods that were used in the experiments.

One of the most known methods used for character recognition is MLP. The MLP, as shown in [1][2], is capable of handling several classes in a single problem and it is well known for its generalization capacity. Another advantage of MLP is that it is able to segment non-linearly separable classes. Given its advantages, MLP is a common choice for the digit recognition task. MLP however has some setbacks. When using the most common training approach, the back-propagation algorithm [2], MLP can easily fall into a region of local minimum, where the training will stop assuming it has achieved an optimal point in the error surface. Another hindrance is defining the best network architecture to solve the problem, considering the number of layers and the number of perceptrons in each hidden layer. Because of these disadvantages, a digit recognizer using the MLP structure may not produce the desired low error rate.

Another widely accepted technique is the use of Support Vector Machines (SVM) [2][4]. SVM is applied to binary problems (problems that contain just two classes). Its objective is to find the separation plane which provides the greatest margin distance between two classes, i.e., the optimal separation plane. This plane is what gives SVM its greater advantage. There are other methods, as MLP itself or Radial Basis Function (RBF) networks [2], which can be used for binary problems [2]; these methods may also operate by finding a separation plane between two classes. These other methods, however, may accept any plane that separates the classes, while SVM looks for the optimal separation plane,

978-1-4577-0653-0/11/$26.00 ©2011 IEEE

which gives its improved generalization capability. Another advantage is that it is capable of moving the entire problem into a representation of greater dimensions, enabling it to separate more complex problems. This is achieved by the use of kernel functions [2][4], such as the RBF or polynomial functions. SVM is normally used in recognition applications as follows: for digits, where there are ten distinct classes, ten SVMs are used. Since SVM is a binary classifier, in order to recognize digits, each of the ten SVMs corresponds to one of the ten possible digits. In this manner, the SVM corresponding to the digit 2, for example, is trained to classify an input as a 2 or as not a 2. The two classes that each SVM then comprises are its respective digit and the remaining digits (which make up the "false" output given by the SVM). Because of this behavior, this SVM is named as a "one against all" kind.

Bellili et al. [3] proposed a hybrid method that employs the use of MLP and SVM. The objective of this method is to further reduce classification error rates, improving the performance of a MLP classifier. This method has the objective of enhancing the performance of a simple MLP classifier. In their experiments involving handwritten digits, it was shown that, when using a MLP, the desired output was, in most cases, present in the two output nodes with greatest values. However, a MLP may not present the best generalization capacity for two classes, considering that, in its training phase, it might have reached a point of local minimum. A SVM, on the other hand, is expected to find the best separation plane between two classes, finding the global minimum of the error surface for the classification. With this information in hand, the aforementioned authors devised a hybrid system which, after submitting a pattern to a MLP, verifies if the two greatest outputs (X and Y) presented a high confusion value in the training stage. If so, the pattern is submitted to a SVM specifically trained to classify classes X and Y. The basic idea is that a MLP may not have been able to separate two classes with the largest separation margin possible. The technique that provides this capability is SVM. Therefore, the hybrid method intends to unify the main advantages of MLP and SVM, enhancing the recognition rate of the application. The number of SVMs can be set as desired, albeit in their tests, the authors include only 5, 10 and 15 SVMs in the hybrid method.

The main steps of the hybrid algorithm's classification phase are:

- Submit the pattern to the MLP;
- X, Y = the two greatest outputs produced by the MLP;
- IF there is a SVM corresponding to the pair (X, Y)

  Classification = output of the SVM(X, Y)
- ELSE

  Classification = output of MLP

The schematic presented in Fig.1 illustrates how the algorithm works.

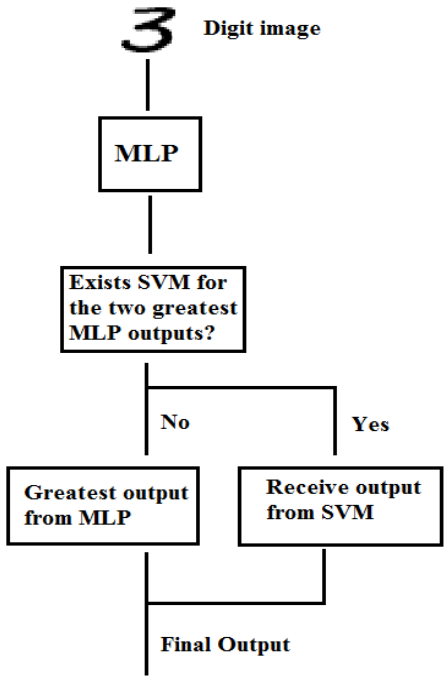

Figure 1. Architecture of Bellili et al's hybrid method.

There are other recognition methods, many of which use the same structures already mentioned, such as MLP or SVM. Ciresan et al. [5] have shown that a MLP can be used to achieve good results for the digit recognition task. They argument that all that is needed is to provide the MLP with enough hidden layers and enough neurons. This alone will yield a classifier of high performance levels. This specific technique has not been explicitly included in the tests because the MLP, its backbone, has already been included. The technique intends to show that, if using a MLP with enough layers and neurons, it can achieve high recognition rates. However, by adding more layers and more neurons, the complexity of the MLP rises greatly. Camastra [6] uses SVMs together with neural gas [7] for recognition purposes. Neural gas is a type of Artificial Neural Network used for vector quantization. The authors propose a method that verifies if the characters are in upper or lower case, and then determine if the upper or lower case forms of a given character can be included into a single class, by using neural gas. The method then submits the characters to the SVM recognizer. These last two methods use a single technique for the purpose of recognition, even though one of them uses neural gas in a pre-processing phase. However, as Bellili et al., there are methods which use more than one technique for the task of recognition. Zhang *et al.* [8] introduced a method which uses an ensemble classifier to classify handwritten digits. It uses gating networks to assemble the outputs of three different neural networks. In this fashion, it relies on a set of classifiers, reaching its output value after analyzing all of its classifiers. Bhowmik et al. [9] proposed a method which used SVMs to classify Bangla characters, which is an Indian script. In this method, a hierarchal SVM classification scheme is used. They also propose the use of neural gas, but, in this case, to produce

grouping schemes for the identification of groups of similar characters. The problem taken on by this proposal is different than the ones already discussed, since the character set it attempts to classify is different. Yet another different application for classifiers is proposed by H. Pirsiavash et al. [10] which combines MLP with SVM for recognition of handwritten Persian/Arabic handwriting. This method also couples principles of fuzzy logic [11] alongside its proposed classifier.

III. HANDWRITTEN DIGIT RECOGNITION PROPOSAL

The proposed method employs only SVM structures. As mentioned before, digit recognizers can be built employing only SVMs. The main difference, however, is how the SVMs are employed in order to build the recognizer.

While analyzing the existing methods, it was evident that the SVMs did, in fact, improve greatly the performance of a MLP classifier. As it is shown in Section IV, the method using MLP coupled with SVMs, as proposed by Bellili et al., improves significantly the performance of a classifier based solely on a MLP. The advantage of adding SVMs to the classifier is evident. However, when analyzing the results of a classifier based solely on SVMs (one against all), also present in Section IV, the results are not as encouraging as expected. Since SVM brings a significant boost to the performance of a classifier using MLP, it would be expected that SVM would have a reasonable performance of its own.

As shown in Section II, usual SVM classifiers employ ten different SVM structures in order to obtain a digit recognizer. Our method, on the other hand, uses 45 SVMs. This change was brought on in attempt to extract a better performance of the classifier based solely on SVMs. The number of SVMs is easily justified by analyzing the scope of the problem. There are ten classes, each class corresponding to a digit (0, 1, 2, 3, 4, 5, 6, 7, 8, and 9). Since the SVM is a binary classifier, the recognition architecture must be designed based on a binary classifier. Therefore, a single SVM is used for each possible pair of digits, which gives a total of 45 pairs, starting on pair (0, 1), (0, 2), (0, 3), up until the pair (8, 9). In this approach, pairs of the same value are not considered, such as (0, 0), since it does not make sense in binary classification. Symmetric pairs, such as (7, 9) and (9, 7), are considered as the same.

To classify a sample, the pattern is submitted to all 45 SVMs. Each SVM then returns a value, which corresponds to the classification it is given for the sample. It is expected that nine outputs indicate the same class, as there are nine SVMs for each class, or digit. For example, for the class 3, there are the following SVMs: (0, 3), (1, 3), (2, 3), (4, 3), (5, 3), (6, 3), (7, 3),(8, 3) and (9, 3). In some cases, however, there can be less than nine outputs indicating a class; that is, a digit 3 can receive only eight outputs indicating that it belongs to the class 3 and still be classified as such.

The main steps of the proposed method are:

1) Training:

- Train 45 SVMs, one for each different pair of digits. Pairs such as (0, 1) and (1, 0) are considered the same.

2) Classification:

- Submit the pattern to all 45 SVMs
- Analyze the outputs of the SVMs

  The class that receives more votes is then chosen as the final output

Fig.2 shows a schematic of the proposed method.

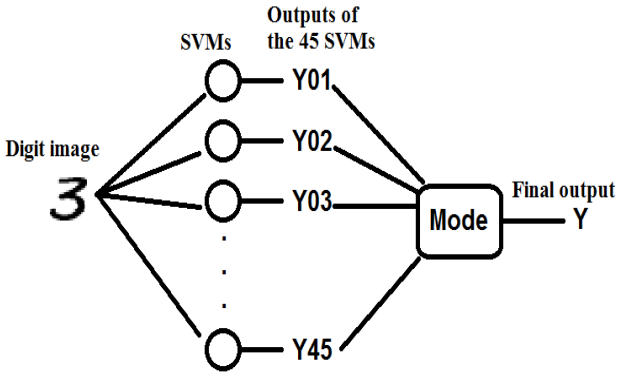

Figure 2. Architecture of the proposed method.

IV. EXPERIMENTS AND RESULTS

This section explains how the different methods mentioned were tested. First, we present the digit database and then the results are presented, along with runtime in seconds for each technique.

A. *Digit Database*

The images were extracted from the NIST SD19 database [12], which is a numerical character database made available by the American National Institute of Standards and Technology. Each image in this database contains a varied number of binarized digits, as it can be seen in Fig. 3.

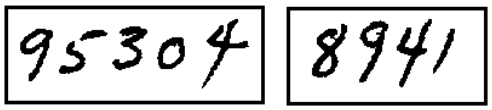

Figure 3. Example of digits from the NIST SD19 database.

The digits were separated by a segmentation algorithm created for this task. This algorithm uses labeling of connected components [13] in order to tag each separate digit and then extract them from the image. The digits after being separated are then manually classified in order to use supervised training methods. The digits found in the NIST database were not all of the same size. Taking this into account, the images containing single digits were normalized according to the following steps: first we establish a 20x25 pixel size as the default size, which

was defined empirically. The next step is to find the coordinates that delimit the digit using vertical and horizontal projections [14]. If the digit possesses dimension greater than 20 X 25 pixels, it is resized to the default size. If the digit has dimension lower than 20 x 25 pixels, the coordinates are ajusted, increasing its width or height, so that the object is in the center of the image with the default size. After these steps, all the digits possess an uniform size (see Fig. 4).

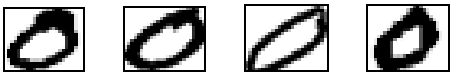

Figure 4. Digit images resized to 20x25.

The complete digit database used in the experiments contains a total of 11,377 digits. Each class contains in average 1,150 digits. This digit database was separated into a training set and a test set. The training set contains 800 digits per class, in average, while the test set contains 350 digits per class, in average. Considering all the classes, the training set contains 7,925 samples and the test set 3,452 samples.

B. Results and analysis

This section presents the results of different digit recognizer techniques. The methods shown here were previously described in Section II. They include classifiers based on: a single MLP, ten SVMs in the "one against all" configuration, the hybrid method using 15 SVMs, the hybrid method using 45 SVMs and our proposed method which relies solely on the use of the 45 SVMs.

The hybrid method, as discussed before, is presented by its authors using 5, 10 and 15 SVMs. In our tests, we included the hybrid method using 45 SVMs. The reason for this is that the more SVMs placed in the hybrid method, the better its performance. Therefore, we aimed to include the methods with best performance in order to properly gauge our own technique.

The algorithms were implemented and tested using Matlab™ [15]. The SVM structure used throughout the tests is the same, changing only how the SVMs are employed and how many of them are used. Therefore, the exact same SVMs, with the same training and parameters, were used for the proposed method and for the hybrid method. The SVMs used for the One Against All method were slightly different, but only in how they were trained and how they separated the classes. Since in this case the SVM considered a digit as one class and all remaining digits as another class, the samples had to be presented to the SVMs in a different fashion than how they were presented to the SVMs in the proposed method. For the SVM, two kernels were used during the training phase: a RBF and a polynomial kernel. The polynomial kernel that yielded the best result was used in the final tests. The same SVM used in our approach was used in the hybrid method. As done with the SVM, the MLP structure used in the tests was the same for both the MLP and hybrid classifiers. It has the following settings:

- An input layer with 500 nodes. Since each digit image was 20x25 in size, its total number of pixels is 500, each node in the input layer corresponds to a pixel in the digit image.
- A hidden layer with 500 nodes. During the experiments this number was reduced and increased but the best result was achieved using this configuration.
- An output layer with 10 nodes, one for each digit.
- Hyperbolic tangent as activation function.
- Gradient descent (back-propagation) as training algorithm.

The results are presented based on the recognition rates of the classifiers, as well as the error rates, as it can be seen in Table I. In Table II, we present the number of misclassifications by each method.

In this test phase, there were two rounds of tests. For the first round, all the test set was employed, containing a total amount of 3,452 samples. The results of these tests are shown in Table I.

TABLE I. RESULTS OF THE DIFFERENT METHODS USING THE TEST SET

Technique	**Results**	
	Error Rate	***Recognition Rate***
MLP	4.37%	95.63%
10 SVMs – One Against All	18.37%	81.63%
Bellili - MLP + 15 SVMs	3.22%	96.78%
Bellili - MLP + 45 SVMs	2.60%	97.40%
Proposed method – 45 SVMs	2.14%	97.94%

As it can be seen, the hybrid method outperforms the MLP classifier. This is due to the fact that sometimes the MLP might not give the correct class as its greatest output. In the majority of the cases the correct class is present in the MLP's two greatest outputs, as stated before. The SVM is capable of selecting the correct class when given two classes to choose from. As the number of SVMs in the hybrid method increases, so does its recognition rate, solidifying the assumption that adding further SVMs enhances the technique. The 10 SVMs in the "one against all" approach achieved the lowest recognition rate of all the techniques, suffering a high 18.37% error rate.

The proposed method, using 45 SVMs, achieved the highest recognition rate, improving the recognition rate of the hybrid classifier at its full potential, which is using 45 SVMs. While analyzing the error rate of the proposed method and the error rate of its closest match, the hybrid method with 45 SVMs, the difference of around 0.5% is seemingly small. But, when one analyzes the real world context in which these applications are inserted, an error difference of 0.5% can produce a far smaller number of misclassifications when hundreds of thousands of digits are being constantly analyzed. Even a small number of errors can be of crucial importance, especially when the cost of an error is extremely high, as is the case with digit recognizers for banks, which use the applications to recognize the values or dates on checks.

Therefore, every gain in performance is considered valuable in this field.

The presented results were obtained, as mentioned before, using the test set, which was extracted from the complete digit set. Another round of tests was made, feeding each method the training set. This different approach was taken in order to measure how well the classifiers were able to classify samples that had already been submitted to them during the training phase. Table II shows the error rate of each method in the training set and in the test set, separately. Therefore, the results in Table II shows how well each of the classifiers learns the training set, as well as how well each of the classifiers is able to generalize from the information it has learned.

TABLE II. RESULTS OF THE DIFFERENT METHODS USING THE TEST AND TRAINING SETS

Technique	**Results**	
	Error rate in training set – 7,925 samples	***Error rate in test set – 3,452 samples***
MLP	2.29%	4.37%
10 SVMs – One Against All	16.82%	18.37%
Bellili - MLP + 15 SVMs	1.62%	3.22%
Bellili - MLP + 45 SVMs	1.17%	2.60%
Proposed method – 45 SVMs	0%	2.06%

As it can be observed, the proposed method shows no misclassifications when considering the training set. The remaining methods, however, show that further misclassifications are committed when regarding the training set. These errors in classifying the training set incurred by the other methods, together with the proposed method´s performance, show the greater learning capacity of the proposal in question.

Another aspect of the behavior of the proposed method that can be analyzed is exactly which digits were misclassified and which digits they were misclassified as. This data yields a confusion matrix, which is shown in Table III. This table is organized according to the target output (the columns) and the actual output (the lines). For instance, if we take the second line, labeled as "1", this represents the misclassifications incurred by the proposed method when it classified a digit as a 1. If we analyze the line labeled as "1", together with the column labeled as "3", we then arrive at a certain position in the matrix. This position gives use the number of times the method's output classified a 3 as being a 1. As it can be seen from Table III, the misclassification with the greatest number of occurrences was the 6 being classified as a 5, yielding five misclassifications.

This table provides important information when attempting to understand exactly where the method commits its mistakes. It is then possible to propose future changes that might further enhance its performance. Some examples of digits that were misclassified can be seen in Fig.5.

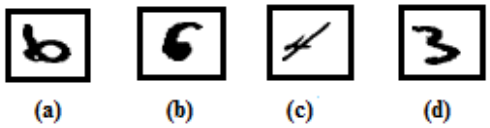

Figure 5. Digit images that were misclassified.

Fig.5a and Fig.5b are examples of the most common type of error committed by the proposed method, which is to misclassify a 6 as a 5, as previously mentioned. Fig.5c and Fig.5d are examples of different types of misclassifications, where a 4 has been classified as a 1 and a 3 has been classified as an 8, respectively.

TABLE III. CONFUSION MATRIX FOR THE PROPOSED METHOD

Output	**Target**									
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	0	0	0	1	0	3	2	0	0	0
1	1	0	0	0	1	0	0	0	0	0
2	3	0	0	1	1	0	0	0	3	1
3	1	0	1	0	0	1	0	1	3	0
4	0	0	0	1	0	0	0	2	1	3
5	2	0	1	1	0	0	5	0	0	1
6	1	0	2	1	0	2	0	0	1	0
7	0	1	0	1	0	0	0	0	0	2
8	0	0	3	1	1	4	0	0	0	1
9	1	0	0	0	2	0	0	4	2	0

It is also important to acknowledge the speed of the proposed method, although it uses 45 SVMs to classify each input, its runtime did not suffer greatly in comparison to the other methods. Although we do not present runtimes as a palpable result, there is still margin for some comments. The algorithms were implemented in Matlab, and therefore, used several of the built-in Matlab functions. No code optimization was used. Even so, when analyzing runtimes for single digits, the proposed method achieved an average of 0.0884 seconds of processing time. Bellilli et al.'s method, with 45 SVMs, recognized a single digit in an average of 0.0340 seconds, while the MLP, took in average 0.0323 seconds to recognize a single digit. The natural impression is that the proposed method takes around twice the time to recognize a digit than the remaining techniques. When the runtimes are measured using the entire digit set however, a different assumption can be made. The proposed method obtained an average 23.914 seconds of runtime, while Bellilli et al.'s method using 45 SVMs obtained a 23.313 second average. If the number of SVMs in Bellili et al.'s method is reduced, so is its runtime. Bellili et al.'s method using 10 SVMs averaged 10.697 seconds of runtime.

V. CONCLUSIONS

This paper presents a method for handwritten digit recognition which is based on the use of 45 Support Vector Machines. Each SVM is set to recognize a pair of digits instead of a single digit. The proposed method achieved its goal, which was to further improve the performance of existing handwritten digit recognizers. The new approach in using SVMs proved valid, since the performance of a classifier using solely SVMs was greatly increase and since it also outperformed MLP and hybrid classifiers. Also, when regarding the performance of the proposal in question when applied to the training set, it can be concluded that if the training set is sufficiently representative of the data, the method can yet increase its performance.

As expected, the MLP provides a high recognition rate but falls behind when compared to newer and more refined techniques.

The hybrid technique proves to be a valid technique in its own right, but it is obvious that this approach is susceptible to the performance of the two techniques: MLP and SVM. If the hybrid classifier has an extremely well trained set of SVMs, but if the MLP itself does not yield satisfactory performance, the whole application suffers. With a poorly performing MLP, the SVMs will not receive the correct pair information. For example, the sample may contain a digit '1', but the two greatest outputs of the MLP are '7' and '8', there is no way for the corresponding SVM to correctly classify the sample as a '1' since the information it receives states that the sample it either a '7' or '8'. In the other hand, if the hybrid method possesses a well-trained MLP and poorly trained SVMs, its performance will not be enhanced to the fullest potential. The proposed method needs only one structure to be well-trained, which is the SVM. With no co-dependencies, the proposed method needs only to concern itself with the management of its SVMs.

While no conclusive time tests could be made, it can be seen that the proposed method does not suffer greatly when compared to its competitors. Some methods may indeed be faster, even twice as fast, but even for a 11,000 digit database, all methods are able to process all the digits well under 30 seconds. The gains in performance brought on by the proposed method can easily overcome its speed deficiency in regards to other less accurate methods.

ACKNOWLEDGMENTS

This research is partially sponsored by CAPES, CNPq under grant 473926/2010-5, FACEPE APQ-0266-1.03/10 and FACEPE IBPG-0460-1.03/09.

REFERENCES

[1] C. M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, Oxford, UK, 1995.

[2] S. Haykin, Neural Networks: A Comprehensive Foundation, 2nd Edition, Prentice Hall, 1999.

[3] A. Bellili, M. Gilloux, and P. Gallinari, "An MLP-SVM combination architecture for offline handwritten digit recognition. Reduction of recognition errors by Support Vector Machines rejection mechanisms", International Journal on Document Analysis and Recognition, vol. 5, pp. 244-252, February 2004.

[4] V. N. Vapnik, Statistical Learning Theory, John Wiley and Sons, New York, USA, 1998.

[5] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep, big, simple neural nets for handwritten digit recognition," in Neural Computation, vol. 22, pp. 3207-3220, December 2010.

[6] F. Camastra, "A SVM-based cursive character recognizer," in Pattern Recognition, vol. 40, pp. 3721-3727, March 2007.

[7] T. Martinez, S. Berkovich, and K. Schulten, "Neural Gas network for vector quantization and its applications to time-series prediction," IEEE Trans. Neural Networks, vol. 4, no. 4, pp. 558-569, July 1993.

[8] P. Zhang, T. D. Bui, and C. Y. Suen, "A novel cascade ensemble classifier system with a high recognition performance on handwritten digits," in Pattern Recognition, vol. 40, pp. 3415-3429, March 2007.

[9] T. K. Bhowmik, P. Ghanty, A. Roy, and S.K. Parui, "SVM-based hierarchal architectures for handwritten Bangla character recognition," International Journal on Document Analysis and Recognition, vol. 12, pp. 97-108, 2009.

[10] H. Parsiavash, R. Mehran, and F. Razzazi, "A robust free size OCR for omni-font Persian/Arabic document using combined MLP/SVM," in Proceedings of Iberoamerican Congress on Pattern Recognition, pp.601-610, 2005

[11] B. Kosko, Neural Networks and Fuzzy Systems, Prentice Hall, 1991.

[12] NIST Special Database 19. Handprinted Forms and Characters Database. Link: http://www.nist.gov/srd/nistsd19.cfm . Accessed in February, 2011.

[13] E.R.Davies, Machine Vision, 3rd Edition, Morgan Kaufmann, 2005.

[14] J. R. Parker, Algorithms for Image Processing and Computer Vision, John Wiley and Sons, 1997.

[15] Mathworks Matlab™ – The language of technical computing. Link: http://www.mathworks.com/products/matlab/