Ada-boosting Extreme Learning Machines for Handwritten Digit and Digit Strings Recognition

Raid Saabni
School of Computer Science Department
Tel-Aviv Yaffo Academic College, Tel-Aviv, Israel
Triangle Research & Development Center
Email : raidsa@mta.ac.il

***Abstract*—Automatic handwriting recognition of digit strings, is of academic and commercial interest. Current algorithms are already quite good at learning to recognize handwritten digits, which enables to use them for sorting letters and reading personal checks. Neural networks are a powerful technology for classification of visual inputs arising from documents, and have been extensively used in many fields due to their ability to approximate complex nonlinear mappings directly from the input sample. Different variations of Multi-Layer Neural Networks (MLNN), using back propagation training algorithm, yield a very high recognition rates on handwritten digits benchmarks, but lacks the aspect of speed in training time. Learning time is an important factor while designing any computational intelligent algorithm for classifications, especially when online improving by adapting new samples is needed. Extreme Learning Machine (ELM) has been proposed as an alternative ANN, which significantly reduce the amount of time needed to train a MLNN and has been widely used for many applications. The ELM analytical process of learning reduces the time of learning comparing to back propagation by avoiding the process of iterative learning. In this paper, we present a process which boosts few Extreme learning machines using Ada-boosting in order to improve the recognition rates iteratively. A pre-processing step is used to improve the ability of the ELM, and special weighting process to improve the boosting process. To evaluate the presented approach, we have used the (HDRC 2013) data-set which have bee used at the 2014 competition on handwritten digit string recognition organized in conjunction with ICFHR2014 of Western Arabic digit string recognition with varying length. Very high accurate results in terms of very low error rates while keeping efficient time of online training were achieved by the presented approach, which enables on demand time/precision tradeoff.**

Keywords*—Extreme Learning Machines, Neural Networks, Handwritten Digit Recognition, Ada-boost

I. Introduction

Automatic handwriting recognition of text, digit strings and single digits, is an important task frequently researched, in academic and commercial projects. Among the wide range of applications, which may get benefit of the developed algorithms, are reading amounts on checks, check vitrification and mail sorting systems. The most widely used benchmark for isolated handwritten digit recognition, is the MNIST database. Recently the CVL Single digit benchmark, was released, as the main data set for the ICDAR2013 Handwritten Digit Recognition Competition. The general problem the task of single handwritten digit recognition faces is the similarity between the different digits, like 1 and 7, 5 and 6, 3 and 8, 9 and 8 etc. The uniqueness and variety in the handwriting of different individuals also influences the formation and appearance of the digits. Recognition of handwritten digit string is more complicated due to varying lengths of the strings and touching digits in different positions and orientations.

Multilayer Neural Networks, were among the first classifiers tested on handwritten digit recognition(HWDR), and complex versions of Artificial Neural Networks(ANN), gave the state of the art results for HWDR. Most of the proposed ANN's include few layers and many artificial neurons (units) per each layer. Most of the presented ANN's have been trained using a variant of the famous Back Propagation algorithm or more complex methods which directly leads to very long time of training. In such case, these ANN's can't be used as on-line adaptive systems which adapts new misclassified instances by users feed-backs. Therefore, attempts went toward more and more complex variants of support vector machines(SVM's) [2] and combinations of neural networks ANN's and SVMs [10]. Convolutional Neural Networks (CNNs) achieved a record-breaking error rate [16] using novel elastic training image deformations. Recent methods, pre-train each hidden CNN layer one by one in an unsupervised fashion, then use supervised learning to achieve a 0.39% error rate [13, 12]. One of the biggest MLP so far [14], also was pre-trained without supervision, then piped its output into another classifier to achieve an error of 1% without domain specific knowledge. In other approach[1],the authors developed a statisticaltopological feature Combination for recognition of Arabic, Bangla, Devanagari, Latin and Telugu numerals using PCA/MPCA based statistical features.

Extreme Learning Machine proposed by Huang *et al.* [8, 9, 7] uses Single Hidden Layer Feed forward Neural Network (SHLFN) architecture. It randomly chooses the input weights and analytically determines the output weights of the SHLFN. This approach, claims to have much better generalization performance with much faster learning speed. It requires less human interventions and can run thousands times faster than those conventional methods. It automatically determines all the network parameters analytically, which avoids trivial human intervention and makes it efficient in online and real time applications.

Training large deep neural networks is hard, as back propagated gradients quickly vanish exponentially in the number of layers [6]. Indeed, previous deep networks successfully trained with back propagation either had few free parameters or used unsupervised, layer-wise pre-training. But in general deep BP-MLPs need more training time considering that online BP for hundreds or thousands of epochs on large MLPs may take weeks or months on standard serial computers.

In this paper, we use several extreme learning machines to train an SHLFN to recognize handwritten digits. The number of hidden neurons and the number of ELM's have been determined using an iterative process and an evaluation process on a validation set. Even though recognition rates of each SHFLN were high, we have treated them as week classifiers and fed them to an Ada-Boost process in order to improve recognition rates. We have tested the system using the MNIST data set which we have separated to training and validation sets, and with CVL[3] Single Digits data set used at the $ICDAR2103$ competition for HWDR. In the second phase, we have expanded the single digits training data set to include single digits segmented manually from touching digits within digit strings. This process is done on strings of digits after a pre-processing step which included enhancement, skew correction and a sliding window for segmentation. After training the system with the extended data set, an algorithm of single digits recognition using a sliding window with varying widths, have been used to sequentially nominate the existence of digits within the digit strings. The rest of this paper is organized as follows: In section II, we describe our approach in details. Experimental results and some directions for future work are presented in sections III and IV.

II. Our Approach

In the presented work, we address the problem of recognizing handwritten digits as a part of a complete system for recognizing handwritten digit strings. The extension of single digit recognition to recognize complete digit strings is done by using sliding windows of different widths, trained on single digits extracted from the CVL digit strings benchmark. The process starts with classifying the image as a single digit image or a complex one including more than one single digit. For a simple image with a single digit, a standard process of single digit recognition is performed. With complex images including more than one digit, we use a sliding windows over the image to detect and recognize potential existence of single digits. Addition to the recognition rates, we have mainly focused on reducing the training time while keeping high recognition rates. The motivation is to enable an on-line training using error feedbacks in order to improve results by adapting new misclassified samples. We have used a modified version of digits images as input patterns which were fed directly to the input layer of the ANN. For each classifier, a single hidden layer MLP have been trained by the Extreme Learning Machine paradigm. We have tested the system with different number of hidden neurons and with different number of classifiers. Following [17] , we have increased the number of training samples by a general set of elastic distortions that vastly expanded the size of the training set. But we have also preprocessed the images before feeding to the input layer to avoid extra distortion and high distances form a perfect template. The adaboost algorithm, have been used to boost the results of each classifier making the next classifier's constraints to achieve better recognition rates by concentrating on the misclassified objects. In addition to the ELM, we have considered two types of neural network architectures for HW digit recognition. The simplest architecture, which is a universal classifier, is a fully connected network with one hidden layer trained using the Back Propagation algorithm. A more complicated architecture we have considered is a convolutional neural network, which has been found to be well-suited for visual document analysis tasks[18]. The traditional ANN trained by gradient based method, have suffered low results of recognition rates and a very high amount of training time. The convolutional approach have been tested and proofed to be a very accurate system but still needed a very high amount of time for training. Therefore we have selected to use the ELM approach. [RAID] The digit strings recognition process starts by classifying the input image to simple or a complex image. This done as in [china] according to the distribution of pixel values using a neural network. A binarized version of the image is used correct the slant and skew of connected components. No segmentation process is used but a sliding window technique is used in order to record a probability of having a digit with a sliding window.

A. Increasing Data-set Variation and Size

The data set of CVL single digits have been extended to include segments of touching digits, in order to enable digit string recognition with unknown lengths. A variety of widths are used to catch the different widths of single digits within digit strings. Before segmentation, we perform image enhancement, binarization using the OTSU method, size normalization based on digits height, skew and slant correction. In the next step, we divide the digit string to single digits by recording pixel values withing the sliding window including the touching strokes in order to be sound with the recognition process. This step is done manually to catch the variety of digits written as touching digits withing the strings. These segmented touching digits are added to isolated single digits already included in the $MNIST$ and CVL data sets.

The images from the MNIST data set have been used without any change in scaling or position. We have converted the color images of CVL single digits data set to gray scale images. Each of these gray-scale images is re-sized to $28\ X\ 28$ pixel images by preserving their aspect ratio, and their center of mass is computed. Each scaled image is positioned by their center of mass in the center of a $32\ X\ 32$ pixel image. All images in both cases have been normalized to the range$[-1, 1]$. Following other projects, where best results on MNIST were obtained by deforming training images in order to increase their number. This process allows training networks with many weights and by that making them insensitive to in-class variability. In the next step, we have followed the approach presented by Patrice *et al.*[17] where they have increased the distribution invariance with respect to elastic deformations corresponding to uncontrolled oscillations of the hand muscles. For example, simple distortions such as translations, rotations, and skewing can be generated by applying affine displacement fields to images. This is done by computing for every pixel a new target location with respect to the original location. The new target location (N_x, N_y) of the original pixel at position (x,y) is given with respect to the previous position by adding the values of $New_x(x, y)$ and $New_y(x, y)$ randomly chosen in a given range $[-R_1, R_1]$. Taking R_1 for example to be 1, we get an elastic deformation which proved to efficiently expand the database and improve the recognition rates[17]. The fields x and y are then convolved with a Gaussian of standard deviation Σ and for controlling the intensity of the deformation we multiply the displacements by a scaling factor. Expanding the data base is an important step in order to enable better generalization by the training process. In order to improve the recognition of each image, we have generated for each image of the size $32\ X\ 32$ a vector with the size 1456 coordinates. these generated vectors were presented to the Extreme Learning Machine input layer for training and for recognition. The Vectors have been generated using a Gaussian filter on the the original image, a reduced image to half size and and a reduced image from the original image to quarter size, which means that we concatenate three images of the same image in three different scales. to each image in each scales two line with horizontal and vertical density histogram were added and a Gaussian filter with $3\ X\ 3$ kernel size was applied on each image. Thus, we got a $32*(32+2)+16*(16+2)+8*(8+2)$ equals to 1456 vector size. See figure 1 for a demonstration of two images of the digit zero (0).

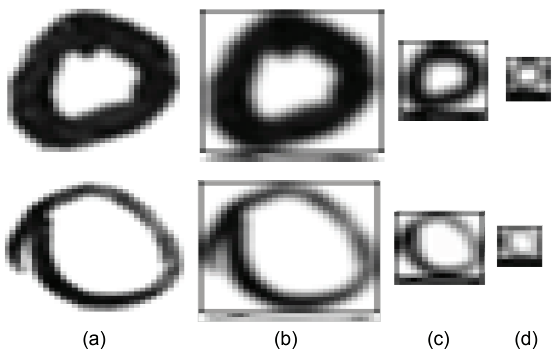

Fig. 1. (a) The two original images of the zero digit (b) the original image in full size have been filtered using a Gaussian filter on a sub window with the size $3\ X\ 3$, (c) and (d) , the reduced image to half and quarter sizes.

B. Extreme Learning Machines

It is clear that the learning speed of feed forward neural networks is in general far slower than required, especially when the training process needs to be on-line and fast. the gradient-based learning algorithms are extensively used to train neural networks and they proved to be very slow due to the iterative process of tuning and changing weight of each neuron to neuron connection for each sample. Unlike these traditional implementations, the authors in [9], proposed a new learning algorithm called Extreme Learning Machine for single-hidden layer feed forward neural networks (SHLFNs). The presented algorithm chooses the input to hidden layer weights randomly. In the next step the algorithm determines the hidden to output weights analytically. Theoretically, this algorithm tends to provide the best generalization performance at extremely fast learning speed. The experimental results based on real-world benchmarking classification problems including large complex applications show that this algorithm can produce best generalization performance in some cases and can learn much faster than traditional popular learning algorithms for feed forward neural networks. By [9], Single Hidden Layer Feed Forward Neural Network (SHLFN) with hidden nodes, can be represented as mathematical description of SHLFN in a unified way as follows:

$$F_l(x) = \sum_{i=1}^{l} \beta_i G(a_i, b_i, x) \tag{1}$$

where the vectors a_i and b_i in the R^n space, are the learning parameters of the hidden nodes. β_i is the weight connecting the $i'th$ hidden node to the output node. When the hidden nodes are additive, $G(a_i, b_i, X)$ is the output of the $i'th$ hidden node calculated using a sigmoid or threshold function of the value $a_i\ .x + a$, and the radial bases function when using (RBF) hidden node type. In the presented approach we have used the additive hidden node with the activation function $g(x) : R \rightarrow R$ given as :

$$G(a_i, b_i, X) = g(a_i \cdot x + b_i,) \tag{2}$$

where a_i is the weight vector connecting the input layer to the $i'th$ hidden node and b_i is the bias of the $i'th$ hidden node. $a_i \cdot x$ denotes the inner product of vector a_i and X in R^n and g is the sigmoid function. For N, arbitrary distinct samples $(x_i, t_i) \in R^n\ X\ R^m$. where x_i is a vector with n coordinate for n Input neurons and t_i is the target vector with m output neurons. if there is a single hidden layer feed forward ANN that can approximate these N samples with no error then their exist a_i , b_i and β_i such that

$$f_L(x_j) = \sum_{i=1}^{l} \beta_i G(a_i, b_i, X), j = 1, \cdots, N \tag{3}$$

which can be written more compactly using matrices as:

$$H\beta = T \tag{4}$$

where,

$$H(a, b, x) = \begin{bmatrix} G(a_1, b_1, x_1) & \cdots & G(a_L, b_L, x_1) \\ \ddots & \vdots & \\ G(a_1, b_1, x_N) & \cdots & G(a_L, b_L, x_N) \end{bmatrix} \tag{5}$$

Since the hidden node parameters of ELM need not be tuned during training and since they are simply assigned with random values, Equation 4, becomes a linear system and the output weights can be estimated as follows.

$$\beta = H^{\dagger} T \tag{6}$$

where $H^{\dagger}$ is the Moore-Penrose generalized inverse of the hidden layer output matrix H. In such case, H can be calculated using several methods including singular value decomposition (SVD) and others. Implementations of ELM uses SVD to calculate the $Moore-Penrose$ generalized inverse of H , since it can be used in all situations which makes the ELM a batch learning method.

C. Boosting the System Using Ada-boost

Boosting is a general method for improving the accuracy of any given learning algorithm. It can be seen as a sophisticated voting process with weights learned manipulating weights of samples depending on their results. The first polynomial time boosting algorithm was presented by Schapire [15]. Later, Freund[4] developed a much more efficient boosting algorithm

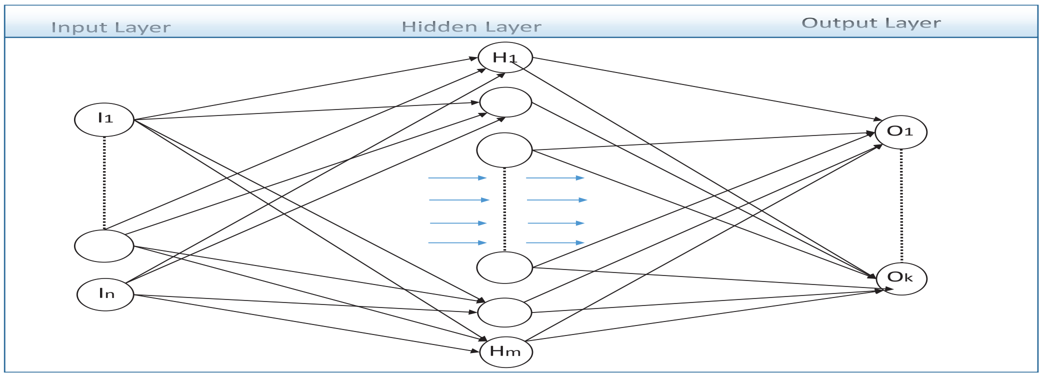

Fig. 2. As shown in this figure, the ELM Neural Network constitute three layers , an input layer with n input neurons,an output layer with k output neurons and a single hidden layer with m hidden neurons. Each layer is fully connected to the next layer

which which was optimal in some certain cases, but suffered from certain practical difficulties. In 1995 Freund and Schapire [5], introduced the Ada-Boost algorithm where they have solved many of the practical difficulties and drawbacks of the boosting algorithms they have presented before. The algorithm takes as input a set of n training pairs $(x_1; y_1) \cdots (x_n; y_n)$ where each x_i belongs to some space X, and each label y_i is a binary label of x_i presenting the labeled target value. The Ada-boosting algorithm, repeat calling a weak learning algorithm, where in each round, it changes the weights of the training samples in such way that the next week classifier will concentrate on the wrong classified samples from the previous round. Formally, if we denote by Dt(i)the distribution on $i'th$ at round t.

The weak learner, seeks to find a weak hypothesis (base classifier) $h_t : X \rightarrow \{-1, 1\}$ appropriate for the distribution D$_t$. The quality of a weak hypothesis is measured by its error

$$\epsilon_t = Pr_{i\ D_t}[h_t(x_i) \neq y_i] = \sum_{i:h_t(x_i)\neq y_i} D_t(i). \qquad (7)$$

Initially, all weights are set equally to be $1/n$. After each round of applying the week classifier on all samples, the algorithm increases the weights of the misclassified examples and decreases the correctly classified ones. The result of this process, forces the weak learner to focus on the hard examples in the training set in such manner that week weighted classifiers complete each other in a sophisticated way that they generate a stronger classifier . We have repeatedly, used the a single hidden layer feed foreword neural network trained by EML as a weak hypothesis. Different instances of the ELM have used different random values in each iteration and led to different results of recognition on the training data set. As part of the Ada-boost algorithm after each ELM iteration we have changed the weight of the training sample in order to make the next ELM concentrate more on the misclassified images. We have tested many configuration including different sizes of hidden layers, different α values during the weighting process and different numbers of week classifiers. The results of the best configuration are presented in section III. It is important to notice that unlike the usual Ada-boost algorithms our week classifiers are not binary classifier by a multi-class ones. see the Algorithm 1. for detailed description.

Algorithm 1 : Ada-Boost Extreme Learning Machine algorithm given a set of n training pairs $(x_1; y_1) \cdots (x_n; y_n)$

```
• Extend the training data-set by deforming images using elastic deformations
• for each Image $I_j$ in the training set
    ◦ for i = 1 to 3
        ▪ Generate horizontal $H_{hist}(I)$ and vertical $V_{hist}(I)$ density histograms.
        ▪ Generate $I'$ by Concatenating to the end of $I$, $H_{hist}(I)$ and $V_{hist}(I)$.
        ▪ Apply a Gaussian filter on $I'$ : $G_{3X3}(I')$ .
        ▪ Convert result to $1 - dimensional$ vector $V_i$
        ▪ Reduce the image to half size.
    ◦ Generate a vector with the size 1456 by concatenating all $V_i's$ : $VI_j = Concatenate(V_1, V_2, V_3)$
• For i = 1 : N (number of week classifiers)
    ◦ Train an $EML$ with $VI_j$ for $J = 1 \cdots n$
    ◦ Weight the samples due to the result of the Ada-boosting process
• return the model of the N classifiers and the $\alpha$ weights from the Ada-Boost process
```

In the recognition process , we start by applying the same image enhancement method, size normalization based on digits height, skew and slant correction as in the segmentation process. On the enhanced image, we use a sliding window with different widths to segment potential digits and fed them as an input to the single digit recognition system. Results of the recognition process, are recorded as probabilities and used to make the final decision of the recognized digit string.

TABLE I. Error rate and time needed for recognition and training of the different configurations on the two benchmarks of MNIST and CVL. The different configuration shows different options of trade off between recognition rate and time for training and recognition which the task can pick due to its constraints. The last two columns, present results of error recognition rates on the CVL string of digits benchmark.

Handwriting Digit Recognizer using Ada-Boost EML						
Number of	MNIST Data Set		CVL Single Digit Data Set		Recognition Rate of Digit Strings	
# of Neurons # of Week Classifiers	Error Rate	Time in seconds	Error Rate	Time in seconds	TOP-3	TOP-1
300 Neurons 5 Week Classifiers	3.92%	35.212 Train 12.307 Recognition	4.16%	12.235 Train 24.078 Recognition	81.08%	72.10%
300 Neurons 15 Week Classifiers	2.31%	124.657 Train 32.436 Recognition	2.47%	39.506 Train 75.106 Recognition	84.33%	78.32%
400 Neurons 5 Week Classifiers	2.65%	37.708 Train 12.008 Recognition	2.68%	12.708 Train 27.006 Recognition	83.12%	77.31%
500 Neurons 15 Week Classifiers	1.55%	157.019 Train 45.207 Recognition	1.67%	61.136 Train 90.716 Recognition	88.01%	82.10%
1500 Neurons 15 Week Classifiers	1.29%	746.237 Train 150.708 Recognition	1.41%	247.231 Train 300.06 Recognition	89.71%	82.96%
2000 Neurons 25 Week Classifiers	1.21%	1877.281 Train 266.217 Recognition	1.29%	627.61 Train 595.12 Recognition	90.02%	83.91 %
2000 Neurons 10 Week Classifiers	1.27 %	667.003 Train 95.203 Recognition	1.38%	292.844 Train 186.708 Recognition	89.01%	73.03%
1000 Neurons 10 Week Classifiers	1.66%	302.713 Train 53.147 Recognition	1.85%	103.056 Train 104.436 Recognition	87.72%	82.21%
5000 Neurons 20 Week Classifiers	1.11%	4723.037 Train 311.228 Recognition	1.23%	1911.06 Train 625.613 Recognition	90.10%	84.01%

III. Data Sets and Experimental results

To Evaluate our system for handwriting digit recognition we have used two standard benchmarks. The first benchmark we have used is the well known MNIST database for handwritten digit recognition. The MNIST data set, is derived from the NIST data set, and has been created by Yann LeCun [11]. The MNIST data set consists of handwritten digit images. The examples for training include $60,000$ examples for all digits and $10,000$ examples for testing. All digit images in this data set have been size-normalized and centered in a fixed size image of $28x28$ pixels. In the original data set each pixel of the image is represented by a value between 0 and 255, where 0 is black, 255 is white and anything in between is a different shade of gray. The second data set is the CVL SINGLE DIGIT DATASET [3], presented as the main data set for the ICDAR2013 Competition on Handwritten Digit Recognition (HDRC 2013). The CVL Single Digit data set is part of the CVL Handwritten Digit database (CVL HDdb), which has been collected mostly among students of the Vienna University of Technology and of an Austrian secondary school; it consists of samples from 303 writers. For the CVL HDdb, 26 different digit strings with varying length were collected from each writer, resulting in a database of $7,800$ samples. In order to create the CVL Single Digit data set, isolated (unconnected) digits were extracted from the CVL HDdb. In the design process of the database, a uniform distribution of the occurrences of each digit was ensured. The images are delivered in original size with a resolution of 300 dpi. Contrary to other data sets, the digits are not size-normalized since in real world cases, differences in a writers handwriting include variation in size as well as writing style. The images for each set were randomly selected from a subset of writers from the CVL Single Digit dataset. The complete CVL Single Digit data set consists of 10 classes (0-9) with 3,578 samples per class. For the HDR competition, 7000 digits (700 digits per class) of 67 writers have been selected as training set. A validation set of equal size has been published with a different set of 60 writers. The evaluation set consists of $2,178$ digits per class resulting in $21,780$ evaluation samples of the remaining 176 writers. We have used two main configuration for testing the system. The first one is the usual case of off-line training and the second is a simulation of on-line training. The two main parameters we evaluated are the training time while the recognition rate kept high, and the on-line training and new instances adaption. for the first configuration the training data set of the MNIST have been separated to two subsets. The first set have been used for training and included $50,000$ samples, while the second set of $10,000$ have been used for validation to help tune the weights of the week classifiers in the boosting process. In the second configuration the MNIST data set have been separated to three sub sets. The first set included 50000 and was used for usual training. The second set of 5000 images was used for validation and the set of the remaining 5000 samples was used as 5 subsets of 1000 images each presented as new instances in the on-line learning process.

The last two columns of the result's table, show error recognition rates on the CVL digit strings benchmark when looking at the TOP-1 and TOP-3 results. In order not, to count any misclassified digit within a digit string as a misclassified sample of all the string, we have used the normalized number of correctly classified digits as the score of classifying the string image. In such case, classifying all the digits within string correctly, counts as one and mistakenly counts as zero.

We have tested the system using different sizes of the hidden layer while measuring the time needed for training and testing, and the recognition rates. The tested sizes were from 300 neurons in the hidden layer to 2000 neurons with steps of 50 neurons each. A special case of 5000 neurons was also tested. The number of the week classifiers we have used was from 5 to 25. The idea behind these different configuration is to try and figure the compact setting which makes the better time/accuracy tradeoff.

As we can see in Table I, results in terms of error rate versus time for training shows that when the factor of time is very important the fourth or the fifth option can be picked and used for on-line learning system. as seen in the fourth line the time needed to retrain the whole system using close

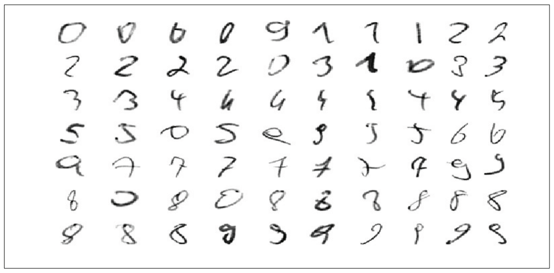
Fig. 3. Some samples of misclassified images. In most cases, the misclassified images were classified by our system as a very similar digit, such that in many cases humans can make such mistakes.

to $20,000$ samples for the CVL data set (extended by elastic deformations) is taking less than 65 seconds on an ordinary home desktop (core $i3$ with 4 GB ROM), while the error rate is only 1.55%, comparing to the last line with 5000 neurons and 20 week classifiers the training process is taking 30 times more for only 0.44 error rate improving. The time needed for recognition in column 3 is for the 10000 samples in the testing set of the MNIST benchmark. The time needed for recognition in column 5 is for the 21780 samples in the testing set of the CVL benchmark. In all configuration when we have used the online configuration adding 1000 samples in each iteration we have adapted the new samples to the system in a very short time and improving the results on the misclassified samples.

IV. Conclusion and future work

We have presented an algorithm that uses Extreme Learning Machine to train a Single Hidden Layer Feed Forward Neural Network for handwritten digit recognition. To enable an iterative learning process and improve results we have used Ada-boost to boost the classifiers based on ELM. The main idea was to keep the training time as low as possible without losing recognition rate. Such approach can easily be used with systems that get users feedback to re-train the system adapting new misclassified samples. The scope of future work includes extending the system to work with digit strings and key word searching in images of handwritten document. another scope of future work, includes using a training process adding neurons to the hidden layer without retrain all the existing connections. Such approach may reduce the training time even more.

References

[1] N. Das, J. M. Reddy, R. Sarkar, S. Basu, M. Kundu, M. Nasipuri, and D. K. Basu. A statistical-topological feature combination for recognition of handwritten numerals. *Applied Soft Computing*, 12:2486–2495, 2012.

[2] D. Decoste and B. Scholkopf. (2002). training invariant support vector machines. *Machine Learning,*, 46:161190., 2002.

[3] Markus Diem, Stefan Fiel, Angelika Garz, Manuel Keglevic, Florian Kleber, and Robert Sablatnig. Icdar2013 competition on handwritten digit recognition (hdrc 2013). In *2013 12th International Conference on Document Analysis and Recognition*, 2013.

[4] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256 – 285, 1995.

[5] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119 – 139, 1997.

[6] G. Hinton. To recognize shapes, first learn to generate images. *Computational neuroscience: Theoretical insights into brain function. Burlington, MA: Elsevier.*, 2007.

[7] Guang-Bin Huang and Chee-Kheong Siew. Lei Chen. Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Transactions on Neural Networks*, 17(4):879–892, 2006.

[8] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: A new learning scheme of feedforward neural networks. In *International Joint Conference on Neural Networks*, volume 2, pages 985–990, 2004.

[9] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70:489–501, 2006.

[10] F. Lauer, C. Suen, and G. Bloch. A trainable feature extractor for handwritten digit recognition. *Pattern Recognition*, 40:18161824., 2007.

[11] Yann LeCun. The mnist database of handwritten digits. http://www.research.att.com/ yann/exdb/mnist, AT&T Labs, 1994.

[12] M. Ranzato, F. Huang, Y. Boureau, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Computer Vision and Pattern Recognition Conference (CVPR07), San Mateo, CA.*, 2007.

[13] M. Ranzato, C. Poultney, S. Chopra, and Y.. In B. LeCun. Efficient learning of sparse representations with an energy-based model. *Advances in neural information processing systems., MIT Press.*, 2006.

[14] R. Salakhutdinov and G. Hinton. Learning a nonlinear embedding by preserving class neighborhood structure. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics. San Francisco: Morgan Kaufmann.*, 2007.

[15] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[16] P. Y. Simard, D. Steinkraus, and J.C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *In Intl. Conf. Document Analysis and Recognition, San Mateo CA: IEEE Computer Society*, page 958962, 2003.

[17] Patrice Y. Simard, Dave Steinkraus, and John C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Proceeding ICDAR '03 Proceedings of the Seventh International Conference on Document Analysis and Recognition*, volume 2, page 958. Institute of Electrical and Electronics Engineers, Inc., August 2003.

[18] L Y. LeCun, Y. Bottou, P. Bengio, and Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278– 2324, 1998.