Recognizing handwritten Single Digits and Digit Strings Using Deep Architecture of Neural Networks

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

***Abstract*—Automatic handwriting recognition of digits and digit strings, are of real interest commercially and as an academic research topic. Recent advances using neural networks and especially deep learning algorithms such as convolutional neural nets present impressive results for single digit recognition. Such results enable developing efficient tools for automatic mail sorting and reading amounts and dates on personal checks. Artificial-Neural-Networks is a powerful technology for classification of visual inputs in many fields due to their ability to approximate complex nonlinear mappings directly from input samples. In this paper we present an approach compromising between the full connectivity of traditional Multi Layer Neural Network trained by Back Propagation and deep architecture. This enables, reasonable training time using a four hidden layers Neural Network and keeps high recognition rates. Pre-trained layers using sparse auto encoders with predefined sequences of training process and rounds, are used to train the net to attain high recognition rates. We have extended the training set to include CVL, $MNIST$ and manually crafted images of single digits from the $ORAND-CAR$ and a private collection of bank checks. Sliding windows technique is used to handle digit strings recognition and obtain encouraging results on CVL and $ORAND-CAR$ benchmarks and our private collection of local bank checks.**

***Keywords*—Handwritten Digit Recognition , Neural Networks ,Deep Learning ,Back-propagation, Sparse auto-encoders**

I. Introduction

In recent years, we witness a huge advance on training deep neural networks for recognition and classification tasks in various fields, such as computer vision and document image analyzing. The $MNIST$ data set is one of the most known benchmarks for testing machine learning techniques, and impressive results were obtained and reported on it. Real tasks of handwritten digit strings, such as real time reading amounts on checks using mobile devices, check vitrification on automatic machines and automatic mail sorting still suffer from low results using the state of the art techniques. This is due to the complexity of the samples, variety of sizes, writing styles and backgrounds from one hand, and the free cursive nature of handwriting in the other hand. The CVL Single digit and the $ORAND-CAR$ [3] digit strings benchmarks, were released at 2013, as the main data set for the ICDAR2013 Handwritten Digit Recognition Competition. The uniqueness and variety in the handwriting of different individuals influences the formation and appearance of the digits and included single digit images and images of digit strings. While recognizing a single digit can be relatively considered as an easy task, recognition of handwritten digit strings on real images of checks is more complicated due to varying lengths of the strings and cursive writing style where touching digits in different positions and orientations occurs frequently.

Multilayer Neural Networks, were among the first classifiers tested on handwritten digit recognition(HWDR)[8, 16], 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 trained using a variant of the famous Back Propagation algorithm. Mostly, using fully connected multi-layer neural nets having large numbers of weights as free parameters to be learned. This approach leads to inconsistent behavior of the learning process, long time for training and over fitting. Other attempts used complex variants of support vector machines(SVM's) [2] and combinations of neural networks ANN's and SVMs [6]. More advanced neural networks have used different deep structures than fully connected layers such as Convolutional Neural Networks (CNNs) to solve such problems. CNNs, achieved a record-breaking error rate [16] by 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 [11, 10]. Another MLP [15], 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 statistical-topological feature Combination for recognition of Arabic, Bangla, Devanagari, Latin and Telugu numerals using PCA/MPCA based statistical features. One of the most famous systems for recognizing handwritten digits is the leNet5[8]. This system uses a convolutional neural network with several hidden layers. Each neuron in the hidden layer is connected to a local area of the input image called the receptive field of that neuron. Overlapping adjacent rectangles of the input image were connected to adjacent neurons in the hidden layer which reserves the local spatial information. The weights of these nodes were trained using the sharing weights paradigm which leads to reducing the number of weights and at the same time, enables extracting the same feature in different places in the previous layer. In the next step a sub-sampling process called pooling used to reduce the layer size and to enable small local translation invariance of feature existence. the final two layers are fully connected layers and all the net is trained using a modified version of the back propagation algorithm.

Training large and deep neural networks is complicated and inconsistent, as back propagated gradients quickly vanish

ISBN: 978-1-4673-9187-0

exponentially in the number of layers [4] and the cost error function suffers odd behavior and many local minimal points. 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, and training process for hundreds or thousands of epochs on large MLPs may take weeks or months on standard desktop computers.

In this paper, we train several sparse auto encoders on single handwritten digits using the k-sparse auto encoder training algorithm[9]. The number of hidden neurons and the level of sparsity is determined using an iteratively based on an evaluation process of a validation set. Results of these Auto-encoders are used for two purposes, the first is to expand the training data set in order to avoid over fitting especially when large numbers of neurons are used, and the second is to use their internal representation as stacked hidden layers in the final neural network. In a pre-processing phase, we have expanded the single digits training data set to include single digits and digits segmented manually from touching digits within digit strings from the $OrandCar$ data set. This process is done on strings of digits after performing image 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. We have trained the system using variants of the available data sets ($MNIST$, CVL, $OrandCar$ and a our private collection of digit strings) and tested it using the $MNIST$ data set, CVL[3] Single Digits and the $OrandCar$ data sets used at the $ICDAR2103$ competition. 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

The presented work is part of a project aims to read data from images of checks. The process starts with localizing the different regions of the data, categorizing them due to their type to figures (bank logo), printed and handwritten text/digit areas. Following the constraints of the CVL and $OrandCar$ data sets containing only images of digits, we present results addressing the tasks of recognizing single hand written digits from the CVL data set, and digit strings from the $OrandCar$ data sets. In the proposed project, we have collected and developed a data set($SDPC$) of 2000 bank checks written by different writers for the various local banks. Curtsey amounts of handwritten digit strings in these checks, have been manually crafted and used for training and testing the proposed approach. Images of digit strings from the $OrandCar$ and $SDPC$ data sets include more than one single digit, in some cases written in cursive style. To process such images, we have used a sliding windows over the image to detect and recognize potential existence of single digits.

To improve results and avoid over fitting, we have expanded the training set by using reconstructed output images of the training set using $k-spare$ auto-encoders. The $k-sparse$ auto-encoder we have used, have a single hidden layer with different numbers of neurons and different values of k. The hidden layers of the selected auto encoders are used as an additional hidden layers in the final neural networks for single digits recognizer. The final recognizer is a four hidden layers fully connected neural network trained and tuned carefully with a variation of the back propagation algorithm. The modification done to the training algorithm in order to enable soft tuning of the weights of each stacked hidden layer separately from the other hidden layers.

Following successful results of [16] and our previous work [13], we used a general set of elastic distortions that vastly expanded the size of the training set. The digit strings recognition process starts by classifying the input image to simple or a complex image according to the distribution of pixel values using a simple neural network. A binarized version of the image is used to correct the slant and skew of connected components. No segmentation process is used but a sliding window technique in order to record a probability of having a digit within a window's area.

A. Increasing Data-set Variation and Size

One of the problems when training large neural networks having many weights as free parameters is the problem of over fitting. While recognition error rate of training continues to go down, it stop and even starts to increase on the test set. Additional to many techniques we have used to prevent over fitting such as validation sets, weight decay and others, extending the size of the training set is one known successful approach. In the presented work we have used few techniques to increase the size of the training data set. The data set of CVL single digits have been extended by all single digit images manually crafted from digit string images. Touching digits were also carefully segmented, in order to extend the data set and 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. In all cases, 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 is a semi-manual step which catches the variety of digits written as touching digits within the strings. These segmented touching digits are added to isolated single digits already included in the $MNIST$ and CVL data sets.

All images have been rescaled to have the same size of $32\ X\ 32$ pixel image preserving their aspect ratio. Images center of mass is computed and each scaled image is positioned by its center of mass in the center. All images in both cases have been normalized to the range$[-1, 1]$. Following other projects, where best results on $MNIST$ were obtained we have used a deformation process on 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 this step, we have followed the approach presented by Patrice *et al.*[16] 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[16]. 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.

Auto-encoders are multi-layer neural networks with at least one hidden layer. In many cases, they are trained using the back propagation algorithm to auto recognize their input by fixing the desired output to be the same as the input, therefore the name auto encoders. Mostly, the result images of the auto-encoders are very close representations of the original input, but still have some variation. When the number of units in the hidden layer is smaller than the size of the input/output layers, activation results on these neurons can be seen as a compressed representation of the input, and as a sparse one when the number is larger. In booth cases, the hidden layer can be seen and used as a feature representation of the given image automatically extracted during the training process. Sparse representation of the image in the hidden layer many times leads to better feature representation. one of the simplest and yet very effective sparse representation using auto-encoders is the k-sparse auto-encoder[9]. Additional effective extension of the training sample in this project is done using the sparse auto-encoders results with different number of hidden nodes and different values of K.

B. Deep Neural Networks

Multilayer neural networks with one hidden layer is easy to use and train. Results in this case on the $MNIST$ data set are impressive and can easily achieve less the 3% error rates on the 50000 samples of the $MNIST$ test set. Unfortunately, these rates are not kept when tasks are more complex and can't be improved; in such case, simply adding additional layers doesn't improve the system. Different approaches to adding layers have to be used in order to improve deep learning systems. In convolutional neural networks(CNN), covolutional/pooling layers with sharing weights are used to add depth to the ANN and by that enable hierarchical feature extraction and representation leading to efficient training algorithm and better recognition rates. Impressive results of CNN's on the $MNIST$ benchmark, were easily adapted to other fields in computer vision. Similar, but still different approach was used by Hinton *et al.*[5] to stack hidden pre-trained layer in unsupervised manner to extract features with restricted Boltzmann machines. In both cases it is obvious that adding layers will be more effective when these layers were first trained a side form the final net. Following this idea, in this project, we have used k-sparse auto encoders with one hidden layers as a first step of feature extraction and representation. A second phase of k-sparse features have been used on the features from the first phase to extract additional hierarchy of feature representation. Hidden nodes from the first step where concatenated to form the first hidden layer and used as an input to the second hidden layer. In the next step we add two fully connected hidden layers with random weights. The final net have been trained and tuned using a slightly modified variant of the back propagation algorithm.

C. k-Sparse Auto-Encoders

Typically in a supervised learning task, a neural network with many layers is initialized with random weights from a Gaussian distribution. Performing back-propagation directly on a freshly initialized network will tend to be very slow and get stuck in poor local minimal points of the loss function as it is highly pathological when parametrized by huge number of free variables with complex dependencies. This leads to deep neural networks that take long time to train and yield poor results[5]. Hinton *et al.*[5] showed that if you pre-train each layer of the network in an unsupervised manner to learn a sparsified representation before the classification task the learning problem is greatly improved.

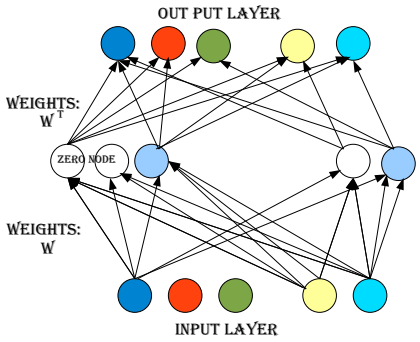

Fig. 1. (a) One hidden layer k-sparse auto encoder. The same colors for input/output nodes to show that auto encoders learn to reconstruct the input image as an output. Empty circle for hidden neurons indicated activations nominated to be converted to zero.

An auto encoder is a neural network with at least one hidden layer that takes as input a feature vector X and learns a code dictionary that changes the input from one representation to hopefully more efficient another one. When the size of the hidden layer is larger than the input and relatively very few nodes are activated, the Auto-encoders is considered as a sparse auto encoder. This factor generally done by a sparsity enforcer that directs the hidden layer to learn a code dictionary that minimizes reconstruction error while restricting the number of activated nodes required for reconstruction, see figure2. The simplest sparse auto encoder consists of a single hidden layer h that is connected to the input by a weight matrix W forming the encoding step. In a tied weights version as in K-Sparse auto encoder the hidden Layer h is connected to the output layer with the matrix weight W^T. The hidden layer then outputs to a reconstruction vector in the output layer where the training process minimize the equation $||X - W^T(Wx + b) + b'||$.

k-sparse auto-encoder[9] is an auto-encoder with one hidden layer, a linear activation function and tied weights. In the feed forward phase, after computing the hidden code $z = Wx + b$ we keep only the k highest value activated nodes to reconstruct the input and zero the rest. Since k is a pre-fixed parameter, we can easily control the sparsity level

Algorithm 1 : The K-Sparse Auto-Encoder training algorithm

- as a feed-forward Compute $z = W^T x + b$
- Sort all activation of z and find the $k - Largest$.
- Set the rest of the nodes to zero.
- Compute the Error $E = ||x - z||^2$
- Back Propagate the error only on the nonzero activations

in the hidden layers. In this case, the selection step acts as a regularizer that prevents the use of an overly large number of hidden units when reconstructing the input. The algorithm for training a k-sparse auto encoder is easy as can be seen in algorithm 1

D. The Deep Neural Network

We have performed two phases of training to generate the final four hidden layers neural network for handwritten digits recognition. In the first phase we have used the training set to train few k-sparse auto encoders for the first hidden layer. We have used large numbers of k in order to learns and extract highly local features involving small stroke and blob detectors. The representation using these local features are fed to another group of k-sparse auto encoder with small k values to capture more global features, see figure2, to see the influence of k values on the extracted features.

as in [9] , The k-sparse auto encoder were used as a pr-trained building block of a deep neural network. We first train a shallow k-sparse auto encoders (two with different k-Values) and obtain the hidden codes as a concatenation of the two. In the next step we fixed the features and train another k-sparse auto-encoder with smaller values of k, and use its hidden layer as an additional block. Next, we use the parameters of the two building block to initialize a four layer deep neural network by adding another two hidden layers initialized by Gaussian random weights. To improve the training process of the full network, a fine tuning process is needed. Makhzani and Brendan[9], first fix the parameters of the first and second layers and train a soft-max classifier on top of the second layer. Next, they train the second layer and soft-max jointly while holding the weights of the first layer fixed and finally, they jointly fine-tune all of the layers with the previous initialization. In our work we have used the same training process with small modifications: 1) in the final training step we have used different learning rates for the different layers, using small values for the pre-trained ones. 2) we have used weight decay and resilient back propagation which in our case lead to faster learning and better recognition rates. We have modified the training process of the auto encoder for better feature extraction. Usually the auto encoders are trained using the same sample as an input and target. In our modification, we have used representative samples of each digit as targets, to trained all samples of that digit. By analyzing the results, we have noticed that this modification made the k-sparse auto encoders learn better features for classification.

Training the neural net is done using the usual back propagation training algorithm with few modification. First we train the Neural net using back propagation while fixing the weights in the first two hidden layers already been trained as

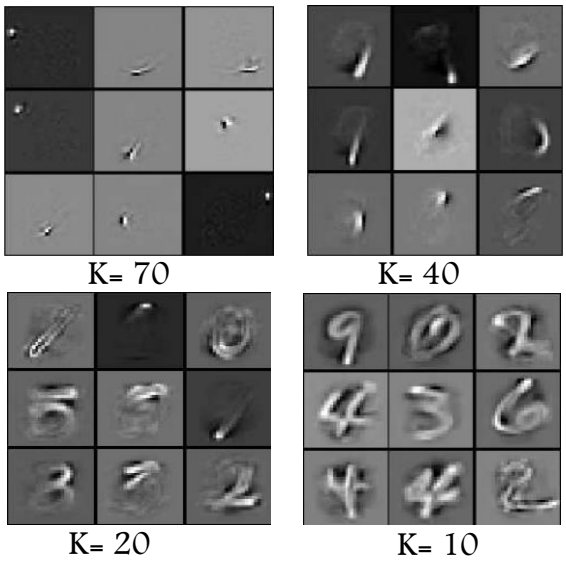

Fig. 2. (a) different values of k show the kind of the extracted features by the k-Sparse auto-encoder[9]. Notice that high values ($k = 70$) for example extracts highly local features while low ($k = 10$) leads to very global features.

part of the auto encoder. In the output layer, Instead of using sigmoid as in all layers, we apply the Soft Max function where,

$$a_j^L = \frac{e^{z_j^L}}{\sum_k e^{z_k^L}} \; where, z_j^L = \sum \sum_k w_{jk}^{L-1} a_k^{l-1} + b_j^L \quad (1)$$

and the log-likelihood cost function $C \equiv -\ln a_y^L$ for a training pair (x, y).

As a very careful and soft tuning process, in a second phase we fix only the first layer and train the last three hidden layer using the soft max function again and different learning rates for the pre-trained layers and the last two hidden layers. We perform the same process again with the first pre-trained layer fixing weights and training the rest. the last step is a full tuning of the deep neural network by the classic mini batch stochastic gradient descent and resilient back propagation (RPROP) using the following equation:

$$w_k \leftarrow w_{k-1} - \eta \frac{\partial C}{\partial w} - \eta \frac{\lambda}{n} \sum_i w_i^2 - \mu \Delta_{k-1} \quad (2)$$

having the learning rate η , the momentum learning rate μ and the weight decay learning rate λ.

Multilayer networks typically use sigmoid transfer functions in the hidden layers. These functions are often called "squashing" functions, since they compress an infinite input range into a finite output range. Sigmoid functions are characterized by the fact that their slope must approach zero as the input gets large. This causes a problem when using steepest descent to train a multilayer network with sigmoid

functions, since the gradient can have a very small magnitude; and therefore, cause small changes in the weights and biases, even though the weights and biases are far from their optimal values.

The idea of Resilient back propagation(RPROP)[12] is to eliminate the harmful effects of the magnitudes of the partial derivatives when using squashing function such as sigmoid and tanh. In the RPROP algorithm, the step size of the weight change is determined by a separate update value for each weight which takes into account only the sign of the partial derivative over all patterns (not the magnitude), as can be seen in equations 3. The updating value of each weight and bias decrease if there was a sign change of the partial derivative of the total error function compared to the last iteration. This is done to the update value for that weight by multiplying by a factor η, where $\eta - < 1$. If the last iteration produced the same sign, the update value is multiplied by a factor of$\eta+$, where$\eta + > 1$. The update values are calculated for each weight in the above manner, and finally each weight is changed by its own update value, in the opposite direction of that weight's partial derivative, so as to minimize the total error function. $\eta+$is empirically set to 1.2 and $\eta-$to0.5.

$$\Delta_{ij}^{(t)} = \begin{cases} \eta^+ \cdot \Delta_{ij}^{(t-1)}, if \frac{\partial E^{(t-1)}}{\partial w_{ij}} \cdot \frac{\partial E^{(t)}}{\partial w_{ij}} > 0 \\ \eta^- \cdot \Delta_{ij}^{(t-1)}, if \frac{\partial E^{(t-1)}}{\partial w_{ij}} \cdot \frac{\partial E^{(t)}}{\partial w_{ij}} < 0 \\ \Delta_{ij}^{(t-1)}, otherwise \end{cases} \quad (3)$$

III. Data Sets and Experimental results

To Evaluate our system for handwriting digit recognition we have used the CVL single digit data set, the $OrandCar$ digits strings and the private collection $SDPC$. The (*$CVLHDdb$*) benchmark was 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 (*$CvlHDdb$*), 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 *$CVLHDdb$*, 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 *$CVLHDdb$*. 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 data set. The complete CVL single digit data set consists of 10 classes with 3578 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 2178 digits per class resulting in 21780 evaluation samples of the remaining 176 writers. The private collection $SDPC$ include 2000 bank checks written by 40 writers. The checks have been selected randomly from bank images of checks and included few types (templates) of check and different writing styles. 2000 handwritten digit string have been manually crafted from these checks and manually separated to single handwritten digit. The final data set of single digit images from the $SDPC$ collection included 5000 images added to the $OrandCar$ data set for training and testing. We have also used the old and well known MNIST database for handwritten digit recognition both for evaluation and as extra sets of training for the CVL task. The MNIST data set, is derived from the NIST data set, and has been created by Yann LeCun [7]. The MNIST data set consists of handwritten digit images. The examples for training include 60000 examples for all digits and 10000 examples for testing. All digit images in this data set have been size-normalized and centered in a fixed size image of 28 X 28 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. We have used two main configuration for testing the system. The first evaluated the system only on the MNIST benchmark. In this evaluation, we have separated the training set to 50000 samples for training, and kept 10000 samples to be used for validation. In the second configuration the MNIST set, the CVL single digits data set and all single digit manually crafted from the $OrandCar$ and $SDPC$ data sets have been used. The 10000 validation set of the MNIST and the validation sets of CVL have been added to 10% samples from the $OrandCar$ to perform as a validation set. The expanding process described above was used to double the size of the training set almost four times. The system have been evaluated using the data sets as is, with the elastic expansion process, and with the reconstructed images using the sparse k-auto encoders.

TABLE I. Error rates for Top-1 and Top-2 recognitions of the system using the different configurations on the MNIST, CVL and ORAND-CAR data sets. The table presents the different configurations error rates and the improvement because of the expansion of the training data set.

Handwriting Digit Recognizer using Deep Learning					
MNIST		CVL Single Digit		$OrandCar$ Digit Strings	
Error Rate Top-1	Error Rate Top-2	Error Rate Top-1	Error Rate Top-2	Error Rate Top-1	Error Rate Top-2
Training Only with original training set					
2.5%	2.0	5.5%	3.7	14.2%	11.1
Training with training set expanded by elastic deformation					
1.6%	1.5	4.0%	2.5	11.6%	8.0
Training with all expanded sets include reconstructed images					
1.0%	0.2	1.5%	1.0	8.3%	6.8

The last two columns in the tableI, show error recognition rates on the $OrandCar$ digit strings benchmark when looking at the TOP-1 and TOP-2 recognized results. In order to take into account partial recognized strings, 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 a string correctly, counts as the number of the digits within the string and one mistaken single digit counts as one, see figure 3 for some examples.

We have tested the system using different sizes of the pre-trained hidden layers using different values of k . In the table we show only the best results we got in several rounds. The results are for a 4 hidden layers neural network with 230 neurons in the first, and 75 in the second hidden layers, and 50 neurons in each of the the next two hidden layers. We have used 65 and 50 as k values for the first k-sparse encoders and 20 for second layer.

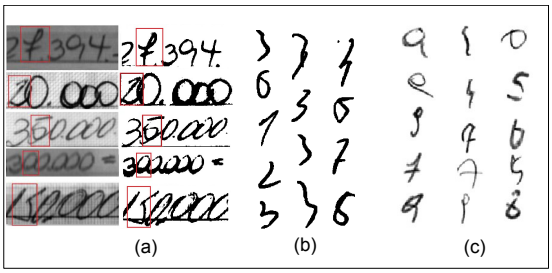

Fig. 3. Some samples of misclassified images. (a) marked inside red rectangles, misclassified single digits within digit strings using the sliding window. (b and c) misclassified examples from the CVL and the $MNIST$ single digits data sets respectively. The images are left without labels in order to show that some of the misclassified digits are not easy to recognize even for human observer.

As we can see in Table I, results in terms of error rate, show that extending the size of the training set improves recognition rates witnessed by the better results in the last rows compared to the first row. Results in all rows in general show that using the architecture of deep neural network based on stacked hidden layers of k-sparse auto encoders can improve results beyond the ability of regular fully connected deep neural network trained by classical back propagation.

IV. Conclusion and Future Work

We have presented an algorithm that trains k-sparse auto encoders and used their hidden layers to be stacked as pre-trained hidden layers into a deep neural network. We have trained the system on an extended version of the MNIST and CVL benchmarks and evaluated it in recognizing single handwritten digits and digit strings from CVL and ORAND-CAR data sets. The proposed system is part of a more complex system aims to analyze images of checks in order to extract and recognize important information such as amounts and texts from checks images. To avoid training deep layer with the back propagation algorithm directly on randomized weights, the first two layers have been trained a side using sparse auto encoders to extract important features in hierarchical manner. The reconstructed images during the training process of auto encoders using different parameters have been used to expand the training set. The scope of future work includes extending the system to work on digit and text strings directly on check images using synthetic string generation [14]. Another scope of future work, includes using modified training process and modules to improve the feature extraction and representation efficiency.

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. Training invariant support vector machines. *Machine Learning,*, 46:161190., 2002.

[3] M. Diem, S. Fiel, A. Garz, M. Keglevic, F. Kleber, and R. Sablatnig. Icdar 2013 competition on handwritten digit recognition (hdrc 2013). In *The 12th Int. Conference on Document Analysis and Recognition (ICDAR)*, pages 1454–1459, 2013.

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

[5] G. Hinton, O. Simon, and T. Darrell. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, July 2006.

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

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

[8] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE.*, 1998.

[9] A. Makhzani and B. Frey. k-sparse autoencoders. *CoRR*, abs/1312.5663, 2013.

[10] 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.

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

[12] M. Riedmiller and H. Braun. Rprop - a fast adaptive learning algorithm. Technical report, Proceedings of the International Symposium on Computer and Information Science VII, 1992.

[13] R. Saabni. Ada-boosting extreme learning machines for handwritten digit and digit strings recognition. In *Digital Information Processing and Communications (ICDIPC), 2015 Fifth International Conference on*, pages 231–236, Oct 2015.

[14] R. Saabni and J. El-Sana. Efficient generation of comprehensive database for online arabic script recognition. In *ICDAR '09: Proceedings of the 2009 10th International Conference on Document Analysis and Recognition*, pages 1231–1235, Washington, DC, USA, 2009. IEEE Computer Society.

[15] 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.

[16] P. Y. Simard, D. Steinkraus, and J. C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, volume 2, pages 958–962. Institute of Electrical and Electronics Engineers, Inc., August 2003.