
# MIT Sloan School of Management

**Working Paper 4365-02**
**May 2002**

## TRAINING NEURAL NETWORKS FOR READING HANDWRITTEN AMOUNTS ON CHECKS

Rafael Palacios and Amar Gupta

# Training Neural Networks for Reading Handwritten Amounts on Checks

RAFAEL PALACIOS* and AMAR GUPTA**
*Escuela Técnica Superior de Ingeniería (ICAI), Instituto de Investigación Tecnológica
*Universidad Pontificia Comillas, Alberto Aguilera 23, E-28015 Madrid, SPAIN
**Sloan School of Management, Massachusetts Institute of Technology,
**77 Massachusetts Ave, Cambridge, MA 02139-4307, USA
Rafael.Palacios@iit.upco.es, agupta@mit.edu

*Abstract:* - While reading handwritten text accurately is a difficult task for computers, the conversion of handwritten papers into digital format is necessary for automatic processing. Since most bank checks are handwritten, the number of checks is very high, and manual processing involves significant expenses, many banks are interested in systems that can read check automatically. This paper presents several approaches to improve the accuracy of neural networks used to read unconstrained numerals in the courtesy amount field of bank checks.

*Keywords:* - Optical character recognition, neural networks, document imaging, check processing, unconstrained handwritten numerals.

## 1 Introduction

Reading Handwritten numerals is an active area of research with practical applications in different fields such as form recognition, postal ZIP codes reading and check reading [2, 7, 14, 15].

Off-line recognition of machine-printed documents has been very successful, and several commercial applications are available. In contrast, handwritten text is more difficult to read by computer systems; such systems are generally slower and yield less accurate results than human beings. Only partial success has been attained by systems for on-line handwriting recognition, since these systems are able to use additional information like the number of strokes, writing speed, direction of curvature, etc [4]. These characteristics are very difficult to obtain after the text has been written on the paper. The most successful systems are those that impose restrictions to the writer, such as preprinted boxes on forms or writing one character at a time in on-line systems.

Nevertheless, tuning recognition systems for specific applications can improve accuracy rates. Tuning can be done by solving particular problems or by improving one performance parameter at the cost of worsening another parameter, based on the assumption that the first parameter is more important for the particular application. In check processing one wants to avoid wrong readings, since the costs associated with incorrect readings are very high. However, it is adequate to reject uncertain checks, which means that the system is unable to read the check with the predefined level of confidence, to help ensure that accepted checks have been read correctly. This paper shows how to improve a system for processing handwritten checks by tuning the module in charge of individual digits recognition. The reduction of the rate for incorrect readings has been attained using complementary approaches.

According to the Federal Reserve Bank, checks account for 60% of the non-cash transactions, and nearly 50 billion checks worth $47.7 trillion were processed in 2001 in the United States alone [5]. Despite the rapid growth of credit and debit cards, at least 12 billion checks are handwritten at the point of sale. But this is not where checks are most commonly used, since more checks are written for bill payment or remittance than for any other purpose (25.7% of check volume). The volume of checks is expected to remain high is spite of the high social costs of paper processing [12]. Humphrey and Berger estimated the social cost of a check transaction to be $0.79 in 1990 [8]. Then Wells used 1993 data to estimate the social cost of a check transaction at $2.78 to $3.09 [16] (both values are higher in 2002 dollars). Since most of the checks need to be partially processed by hand, there is a significant interest in the banking industry for new approaches that can read paper checks automatically, both to reduce the costs as well as to accelerate the check processing operations.

## 2 Description of the System

Our approach for automated check reading is summarized in Fig.1 [10]. This figure describes the key steps, including feedback loop for segmentation.

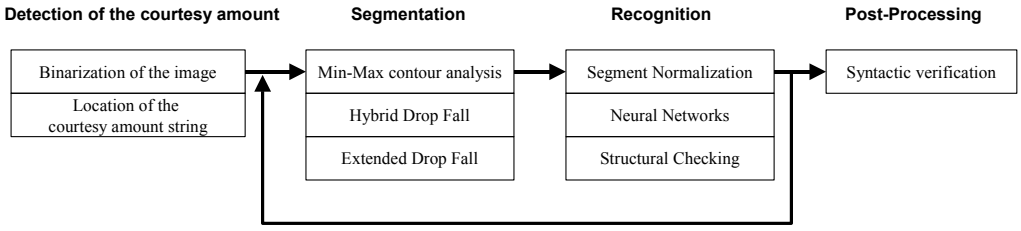


Fig.1: Scheme of the check recognition approach

The first step in the process is to detect the courtesy amount within the image of the check. This involves a conversion from the gray scale image into a binary data format. Then, several algorithms are applied to accurately select the area of the image that corresponds to the courtesy amount field. The most challenging part of the process is the segmentation process, which involves dissecting the courtesy amount field into individual characters. The latter task is performed using a feedback mechanism that helps to determine if the sets of segments, produced by different dividing methods, have been correctly recognized. The recognition module uses a combination of neural networks and structural techniques to classify digits with very high levels of confidence. The final post-processing module verifies the syntax of the amount to minimize the instances involving incorrect readings. This module verifies that the amount read makes sense as a valid monetary value, according to the rules that define the format of numbers (the number of decimals, and the number of digits between punctuation).

## 3 Special Problems Related to Courtesy Amounts in Checks

Checks present the full challenge of totally unconstrained writing because they were not designed for processing by document understanding systems. Segmentation of the string that contains the amount of the check into individual digits is the most difficult task of check processing. This process may involve the separation of touching characters, and the merging of character fragments with other pieces. The approach described in the previous section is able to read amounts containing multiple segments (segments comprised of more than one digit). The recognition module is not able to classify a symbol that is not a digit, so multiple segments are sent back to the segmentation module in the feedback look. Then, the segmentation module has the chance to apply additional dividing algorithms to separate the multiple into two digits. Several dividing algorithms are used at this point; as such, the system is very likely to find the correct dividing path [9]. It is important to note that the feedback process requires the recognition module to reject multiple segments. This is one the issues analyzed in the following section.

Another important characteristic of bank checks is that courtesy amount is more than a sequence of digits, as individuals tend to use other symbols for writing monetary amounts. With respect to U.S. checks, a finite state automaton was proposed in [1] and [6] to segment and analyze the variety of styles found to express cents in fraction format. In other countries, the decimal part of the amount is never found in a style other than scientific format, but sometimes delimiters are used as suffixes or prefixes of the amounts. A few examples taken from Brazilian checks are shown in Fig.2.

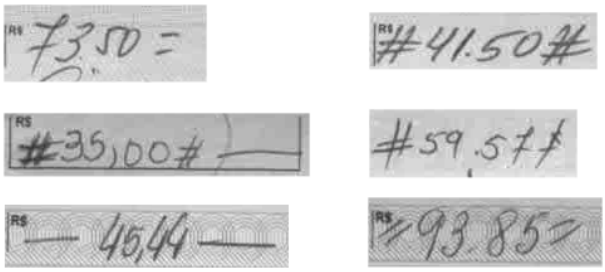

Fig.2: Use of delimiters in Brazilian checks

In a study made with 1500 real checks provided by Brazilian banks, delimiters were found in 36% of the checks, and multiple segments in 20% of those checks.

## 4 Recognition of Isolated Digits

Segmented digits are recognized using a classifier base on Neural Networks. Before using the neural network, the segments are corrected in slant and then normalized in size and thickness [9].

The structure selected for this module is the Multi-Layer Perceptron (MLP), which is one of the most widely used types of networks for character recognition. The basic structure used is a fully

connected MLP with 117 inputs, one hidden layer with 50 neurons and 10 outputs. Its scheme is depicted in Fig.3, where every connection between two layers is represented in terms of the weights and biases applied to each neuron in the former layer to each neuron in the later layer.

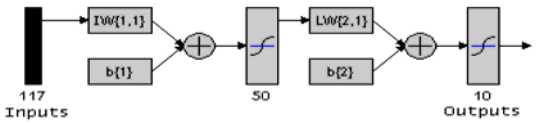


Fig.3 Basic Scheme of MLP

In this case, input weights (IW) are stored in a 50x117 matrix, layer weights (LW) are stored in a 10x50 matrix, bias vector_1 consists of 50 values and bias vector_2 has 10 values. As a consequence, the total number of parameters for this network, with 117 inputs and 10 outputs, comes out to be 6410. The basic model fitting theory suggests that more that 6410 samples are needed to train this neural network; otherwise the model will have more variables than equations.

The inputs are 117 binary values, which correspond to each of the pixels in the normalized 13x9 image. Since these values are binary, they could be represented just by one value each. But in that case, the value would need to be coded using 117 bits or 15 bytes. The former data type is not available in the current generation of computers. Accordingly, it was decided to consider the input to the neural network to comprise of 117 independent variables. Moreover, similar values in a continuous variable of this kind do not correspond to similar images in terms of character shapes; as such, the summarization of all pixels into just one value does not offer any real advantage. Nevertheless, dividing the normalized image into smaller regions and coding each region into integer variables may reduce the number of inputs while maintaining a good resolution. The latter approach is under study and the results will be documented in a future paper.

Neural Networks are characterized by their great power of generalization, specially at modeling non-linear behavior. In order to improve the generalization power, the early stopping approach was adopted. This approach is based on using three independent sets of samples: the training set, the validation set and the test set. The training set is used as the main pieces of information to guide the optimization algorithm to adjust the parameters of the network. On each iteration of the optimization algorithm, the error of the validation set is monitored. The validation error should decrease during the training process along with the error of the training set, evaluated in the objective function. But if the optimization algorithm changes the parameters of the model to reduce the error level in the training set in such a way that the validation error increases, this means that the model is being "overfit" to the training data and the generalization power is being diminished. At this point it is necessary to stop the process. Training the neural network with the early stopping approach can be expressed as the optimization problem: "Minimize: training_error, subject to: validation_error decreases at every step". Defining $t_i$ as the expected output vector for sample $i$ out of $n$ samples in the training set, and $\hat{t}_i$ the prediction of the neural network, the error of the training set, defined as the sum of squared error, will be $SSE_{trs} = \sum_{i=1}^{n} (t_i - \hat{t}_i)^2$. Similarly, the error of the validation set can be defined as $SSE_{vs} = \sum_{i=1}^{n} (v_i - \hat{v}_i)^2$. Therefore, the optimization process can be expressed as:

$$\text{Minimize}: \quad SSE_{trs}$$

$$\text{Subject to}: \quad SSE_{vs}(epoch\ j) \le SSE_{vs}(epoch\ j-1)$$

Early stopping is important to obtain generalization power in the network, but it is essential if the training set is not perfect. While the number of imperfect digits is small, the training process will begin to adjust its parameters to solve normal characters because this is the most efficient way to decrease the objective function. Then the optimization algorithm will try to learn unusual characters to reduce the objective function even further. It is at this final step that overtraining occurs, since the model is beginning to learn incorrect images. As an example of imperfect training set, Fig.4 show some samples from NIST database 19, HSF_9, that were incorrectly segmented and then classified as digit '0' in directory `by-class\30` (samples 397, 954, 1740, 2294 and 3097).

Fig.4 Segments from NIST database 19, classified as digit '0' in the CD.

### 4.1 Analysis of Neural Network Outputs

A neural network classifier is expected to produce an output vector where the value that corresponds to the correct character is high and all other values are

negligible. While the real behavior depends on the transfer functions used at the output layer, the value related to the correct digit is usually lower (meaning less-than-perfect confidence), and the values related to other symbols are greater that zero (meaning that the segment matches the shape of other symbols in some features). The module in charge of the output vector analysis is called Solution Selector. This function is designed to choose the correct value from the output vector even if the behavior of the network is not perfect.

A process to select a solution is to find the maximum output of the network. This is not a good approach for the segmentation strategy described above because the classifier never produces `NOT_RECOGNIZED` results, which are essential for segmentation. Another problem in the analysis of the output vector is the occurrence of two or more symbols with similar confidence values, meaning that the segment has features common to both symbols and therefore making a decision is not straightforward.

Solution Selector has been defined to reject those cases where the confidence is not high enough, as well as those cases where the most likely digits have similar confidence values. Two parameters or thresholds are used to regulate the behavior of the analysis function: $a1$ that defines the minimum confidence to accept a segment, and $a2$ that defines the minimum difference between the maximum value and the second maximum value of the output vector.

The behavior of Solution Selector can be expressed in terms of the following heuristic rules:

**Rule 1**:
if max(*output*) < $a1$
then `NOT_RECOGNIZED`

**Rule 2**:
if max(*output*) * $a2$ < secondmax(*output*)
then `NOT_RECOGNIZED`

By adjusting both parameter levels, it is possible to conduct the results of the classifier, and basically mandate the proportion of rejected symbols versus wrong recognitions. The process of setting these parameters properly is described in a subsequent section entitled "Tuning the Classifier for Check Recognition".

### 4.2 Using Concurrent Networks to Improve Accuracy

Two networks of the same structure trained with the same exact data will result in different parameters if the initial weights are assigned randomly. As such, it is possible to have several networks that in theory classify in the same way, but in practice produce slightly different results. By running multiple classification systems in parallel, one can increase the accuracy of the classifier [2,3,13]. As such, the recognition module was implemented as an array of 3 neural networks of the same type working in parallel. The function in charge of comparing the outputs of the networks, called the Arbiter, can be designed according to different heuristics. This implementation of concurrent networks does not necessarily improve the correct recognition rates, but it converts many wrong recognitions into `NOT_RECOGNIZED`; the latter is preferred in check recognition to avoid incorrect readings. The structure of the system with parallel networks is shown in Fig.5.

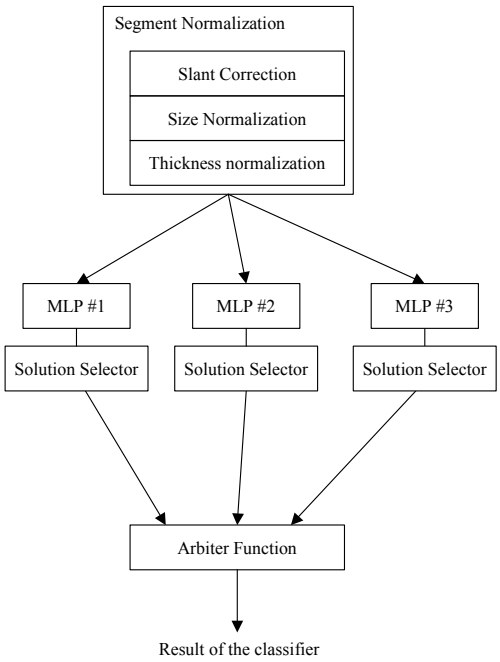


Fig.5: General Scheme of Recognition Module

The results of this architecture are shown in Table 1. The table shows the results obtained with three neural networks that where trained using the same training set and validation set, but initialized with random parameters. A total of 40000 digits selected randomly from the NIST database were used: 20000 training samples, 10000 validation samples and 10000 testing samples. The results of the Solution Selector were obtained using $a1$=0.6 and $a2$=0.8 for the testing set.

Table 1: Performance of multiple MLP neural networks working in parallel

| | samples | Correct | Rejected | Wrong |
|---|---|---|---|---|
| MLP1 | 10000 | 9244<br>92.4% | 570<br>5.7% | 186<br>1.9% |
| MLP2 | 10000 | 9256<br>92.6% | 520<br>5.2% | 224<br>2.2% |
| MLP3 | 10000 | 9213<br>92.1% | 576<br>5.8% | 211<br>2.1% |
| Arbiter | 10000 | 9317<br>93.2% | 525<br>5.2% | 158<br>1.6% |
| Arbiter100 | 10000 | 8788<br>87.9% | 1137<br>11.4% | 75<br>0.8% |

The statistical results show that the behavior is not exactly the same for all three networks. The first Arbiter function was designed to look for the same result in 2 out the 3 networks. The rates obtained in this case, based on outputs from the 3 networks, are better than the results obtained by any one of the networks alone. Arbiter100 is a different method of combining the results of the three networks that looks for 100% coincidence in the results and rejects the solution in any other situation. In this case the number of incorrect reads diminishes drastically; while the number of rejections is increased.

# 5 Tuning Classifier for Check Recognition

Several possibilities exist to enhance the performance of the classifier, by specifically tuning it for purposes of reading courtesy amounts on bank checks.

## 5.1 Parameters of Solution Selector

The first option is to tune the two parameters $a1$ and $a2$ of Solution Selector. Depending on these parameters, the classifier can accept a segment when its confidence is not very high; or it can make rejections when the slighted doubt exists. The desired behavior of the network depends on the nature of the application, and should be selected accordingly. All possible behaviors of the classifier can be represented in a multi–objective graph [11], such as the one shown in Figure 6. This graph represents the percentage of rejections (or `NOT_RECOGNIZED`) versus the percentage of incorrect reads, for different values of the parameters. Each dot in the graph represents the behavior of the neural network for a given pair of parameters $a1$ and $a2$.

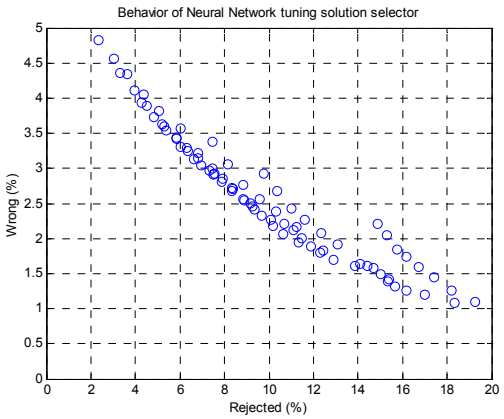


Fig.6: Behavior of classifier for different values of $a1$ and $a2$ in Solution Selector.

Applications to process bank checks require a very low percentage of incorrect readings. Consequently, the systems in the lower area of the graph are better that those in the left side, for this particular application. Nevertheless, Fig.6 shows different behaviors for one neural network alone; in this configuration a small level of wrong reads is allowed since using several networks in parallel will improve the final results, as described above.

## 5.2 Training Neural Network to Recognize Special Characters

The second option for tuning the classifier involves the use of special characters in the training phase. The courtesy amount field on checks frequently includes special symbols in front or behind the value; these symbols are used as delimiters to avoid alterations of the amount. In addition it is common to find digits connected to other digits or delimiters that will appear as one big symbol in the system. In the latter case, more sophisticated segmentation algorithms are needed to separate multiple characters into individual digits. The check recognition system developed by our research group is able to sort out segments consisting of several characters (called multiple segments) at early processing stages and send them to a special function for additional segmentation. But in some cases, multiple segments are sent to the recognition module to try to classify them as digits. At this point, the neural network is supposed to reject these segments since their images do not coincide with any normal digit. However, the neural network is not as effective at rejecting unknown segments as it is at classifying them to the most similar cluster.

After extensive effort analyzing the ability of the neural networks to reject multiple segments or

fragments of digits, it was found that training the network simultaneously with normal digits and special segments produced the best results. Table 2 shows the results obtained by one neural network with Solution Selector using $a1$=0.6 and $a2$=0.8. In the first case, the network (net10) was trained using 20000 digits from NIST database. Then it was tested with normal digits from NIST database, multiple segments obtained from Brazilian and U.S. checks and delimiters from Brazilian checks. The second network (net11) has 11 outputs and was trained simultaneously with digits and multiple segments. The third network (net12) was trained with digits, multiples and delimiters. The number of samples of multiple segments is 246, and the number of delimiters is 286. Both groups of samples were divided into a training set, a validation set and a testing set. The evaluation of the different neural network was performed with the same testing sets. Other researchers have built larger databases of multiple segments, taken from forms images [17], but no database of delimiters was found in the literature.

Table2: Results of training with different sets of segments

| | | samples | Correct | Reject | Wrong |
|---|---|---|---|---|---|
| net10 | NIST Digits | 10000 | 9256 (92.6%) | 520 (5.2%) | 224 (2.2%) |
| | Multiples | 64 | ----- | 31 (48.4%) | 33 (51.6%) |
| | Delimiters | 72 | ----- | 21 (29.2%) | 51 (70.8%) |
| net11 | NIST Digits | 10000 | 9217 (92.2%) | 578 (5.8%) | 205 (2.0%) |
| | Multiples | 64 | 36 (56.2%) | 23 (35.9%) | 5 (7.8%) |
| | Delimiters | 72 | ----- | 36 (50.0%) | 36 (50.0%) |
| net12 | NIST Digits | 10000 | 9140 (91.4%) | 662 (6.6%) | 198 (2.0%) |
| | Multiples | 64 | 38 (59.4%) | 22 (34.4%) | 4 (6.2%) |
| | Delimiters | 72 | 14 (19.4%) | 34 (47.2%) | 24 (33.3%) |

Net10 is able to read the majority of U.S. checks, since this network correctly rejects most of the multiples, and the use of delimiters is rare and negligible in U.S. checks. Nevertheless, net11 does a better job recognizing multiples, even though the number of samples of multiples segments is very low compared to the number of samples of normal digits. By detecting multiple segments, the system is able to apply splitting algorithms [10] and eventually find the correct solution. On the other hand if the multiple segments are misread as one digit, an incorrect read occurs. Fortunately, the post-processing module may detect this kind of mistakes and reject the check at the final step.

In order to read Brazilian checks, the best network structure to use is net12 which is trained to recognize normal digits, multiples and delimiters. The safest approach to read bank checks is to reject all the strings with any `NOT_RECOGNIZED` segment. However, it is possible to accept those amounts where the first or the last segments are rejected, considering that those segments are delimiters. This approach allows one to use net10 and net11 for Brazilian checks; but there are two drawbacks in this approach. First, the risk that one may accept an amount where the first or last digits are rejected and then interpreted as delimiters. Second, most delimiters are classified as normal digits (70% in net10 and 50% in net11). This dilemma can be solved through the use of a neural network able to recognize all different symbols. By using net12, the number of delimiters read as other digits is diminished to 33%, and the likelihood of producing mistakes is lower. Nevertheless, many delimiters can be classified by size and aspect ratio, so only 385 out of 756 delimiters need to be analyzed by the neural network.

# 6 Conclusion

This paper has proposed several ways to improve the recognition rates of systems designed to read bank checks. The use of concurrent neural networks is a powerful technique to help reduce the incidence of incorrect reading of digits at the cost of increasing the number of rejections.

Most of the discussion was focused on how to improve the accuracy of the neural network. One method involved the decision about how to find the correct balance of rejections and wrong readings; such a balance depends on the nature of application. Experimental results show that significant improvement can be obtained by using training sets other than pure digits. This is important for applications where segmentation is difficult or where other non-digit characters can be present. Since delimiters are more similar to normal digits that multiples segments, the improvement obtained after using just a few delimiters during training is not as remarkable as results of using multiples

segments. Using a training set that comprises a better balance of normal digits and delimiters will provide better results.

*References:*


[1] A. Agarwal, A. Gupta, K. Hussein, "Bank Check Analysis and Recognition by Computers" in *Handbook of Character Recognition and Document Image Analysis*. Editors: H.Bunke, and P.S.P. Wang. World Scientific, 1997.

[2] N. Arica and F.T. Yarman-Vural, "An overview of Character Recognition Focused on Off-Line Handwritting", *IEEE Transactions on Systems, Man, and Cybernetics – PartC: Applications and Review*s, 31(2):216-233, 2001.

[3] A.S. Atukorale, P.N. Suganthan, "Combining Classifiers Based on Confidence Values", *Proceedings of the 5th International Conference on Document Analysis and Recognition*, pp.37-40, 1999.

[4] H.Bunke and P.S.P.Wand. *Handbook of character recognition and document image analysis*. Ed. Wolrd Scientific. ISBN 981-02-2270-X, 1997.

[5] Federal Reserve Board, "Fed Announces Results of Study of the Payments System. First Authoritative Study in 20 Years". *Press Release*, November 14th, 2001.

[6] A. Gupta, M.V. Nagendraprasad, P.S.P. Wang, "*System and method for character recognition with normalization"*, U.S. Patent No. 5633954, 1997.

[7] I. Guyon and P.S.P. Wang. *Advances in Pattern Recognition Systems using Neural Network Technologies*. Ed World Scientific. ISBN 981-02-144-8. 1993.

[8] D.B. Humphrey and A. N. Berger. "Market Failure and Resource Use: Economic Incentives to Use Different Payment Instruments" *In The U.S. Payment System: Efficiency, Risk and the Role of the Federal Reserve*, pp. 45–86. Boston, MA: Kluwer Academic Publishers. 1990.

[9] R. Palacios and A. Gupta, "Reading Courtesy Amounts on Handwritten Paper Checks", *submitted to the Journal of Electronic Imaging*, Apr, 2002.

[10] R. Palacios and A. Gupta, "A system for processing handwritten bank checks automatically", *submitted to Image and Vision Computing*, Feb 2002.

[11] R. Palacios, *Modelos especializados en la detección incipiente de fallos*. Tesis Doctoral. Universidad Pontificia Comillas. Julio 1998.

[12] J. Stavins, "A Comparison of Social Costs and Benefits of Paper Check Presentment and ECP with Truncation", *New England Economic Review*, July/August 1997.

[13] C.Y. Suen, C. Nadal, R. Legault, T. Mai, and L. Lam, "Computer Recognition of Unconstrained Handwritten Numerals", *Proceedings of the IEEE*, 80(7):1162-1180, 1992.

[14] C.Y. Suen, Q. Xu, and L.Lam, "Automatic Recognition of Handwritten data on cheques – Fact or Fiction?", *Pattern Recognition Letters*. 20:1287-1295, 1999.

[15] P.S.P. Wang. *Character and Handwriting Recognition: Expanding Frontiers*. Ed. World Scientific. ISBN 981-02-0710-7, 1991.

[16] K.E. Wells, "Are Checks Overused?" *Federal Reserve Bank of Minneapolis Quarterly Review (Fall)*, pp. 2–12, 1996.

[17] J. Zhou, A. Krzyzak, C.Y. Suen, "Verification-a method of enhancing the recognizers of isolated and touching handwritten numerals", *Pattern Recognition*, 35, pp.1179-1189, 2002.