

# From isolated handwritten characters to fields recognition : There's many a slip twixt cup and lip

Christopher Kermorvant<sup>a</sup>, Anne-Laure Bianne<sup>a,b</sup>, Patrick Marty<sup>a</sup>, Farès Menasri<sup>a</sup>  
<sup>a</sup>A2iA Paris

40 bis rue Fabert, 75007 Paris, France

<sup>b</sup>TELECOM Paris Tech/TSI

46 rue Barrault F-75634 Paris Cedex 13, France

{ck,alb,pma,fm}@a2ia.com

## Abstract

*Recognition of handwritten characters has been a popular task for the evaluation of classification algorithms for many years. Looking at the latest results on databases such as USPS or MNIST, one could think that character recognition is a solved problem. In this paper, we claim that this is not the case for two reasons : first because the classical databases for digit recognition are realistic but too simple and second because digit recognition is not a real-world task but only a part of it. In this paper, we contribute to a better understanding of these two aspects with new results. In a first part, we compare three state-of-the-art recognizers on a digit recognition task extracted from a real world application and show that the error rates on this database can not be extrapolated from MNIST. Then, in a second part, we present and evaluate a system designed for an industrial application based on character recognition : document identification with floating field recognition.*

## 1. Introduction

Recognition of handwritten digits or characters has been used as a task for the evaluation of classification algorithms in Machine Learning for many years. Several aspects have made digit recognition a popular task amongst machine learning researchers. First, large databases of digit images have been provided to the community, for example USPS[5], MNIST[7] or RIMES[4]. Moreover, the splitting in train and test data is usually provided, which makes the classification results directly comparable. Second, the nature of the classification problem is challenging for researchers : the variability of writing is large and the dimensionality of the problem is high if one considers pixels as features. Finally, the problem is important since it has many

industrial applications : bank check processing, postal address recognition, automatic invoice processing, etc.

In fifteen years of research, great progress has been made and the latest results on handwritten digit recognition databases such as MNIST or RIMES seem to suggest that the problem is solved : 0.4% error rate on MNIST [8], less than 1% error rate on RIMES[4] digit recognition task. However, we think that these results are misleading. First, since these databases have been used in experiments for many years, one can suspect that a hill-climbing effect has occurred so that the current best classification algorithms and their best parameters might be specific to these databases. Second, thinking that digit recognition is a real-world task is incorrect. In real applications, digit recognition is only one part of the complete recognition process, amongst document layout analysis, orientation correction, skew and slant correction, field segmentation, rejection of non-characters, etc. It is well known that these steps must be optimized all together in order to build a complete industrial application such as a bank check recognizer [3].

In this paper, we contribute to a better understanding of these two aspects with new results. First we evaluate three state-of-the-art character recognition systems on a database of character images extracted from an industrial application and show that results previously published on USPS or MNIST can not be extrapolated. Second we present the architecture and the performance of a recognizer dedicated to a real-word task based on character recognition, task we call floating field recognition. We show that several aspects, other than the error rate on isolated characters, have a major impact on the performance of the system : the localization of the field, its syntactical structure, the presence of a control key and the confidence score associated to the recognition result.

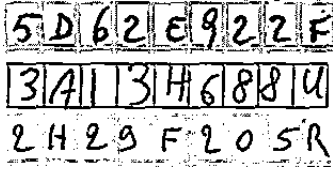


Figure 1. Examples for the A2iA-AV database.

## 2 Handwritten digits recognition : from realistic data to real data

In this section, we present a database extracted from a real-world application and evaluate three state-of-the-art classifiers on a digit recognition task both on a realistic but clean database (MNIST) and on our noisy database (A2iA-AV-Digit).

### 2.1 MNIST and A2iA-AV databases

The MNIST database [7] is a very popular database of isolated handwritten digits used for the evaluation of many machine learning algorithms. The database is composed of 60 000 examples in the training set and 10 000 in the test set. Simard *et al.* [8] have shown that the performance of a digit classifier can be improved by increasing the amount of training data with elastic distortion of the original data. They have proposed an algorithm to generate artificially distorted data. We have extended the MNIST train set by applying 4 random transformations to each one of the 60 000 images. The resulting training set is composed of 300 000 snippets and is referred to as the *Extended-MNIST* database.

We have collected a database of images of Italian fiscal forms. These forms contain several alpha-numerical fields: name, address, dates, amounts, fiscal code. We have manually annotated 36 337 forms with the location and the value of the fiscal code field. We refer to this database as A2iA-AV-Field. The code is composed of 16 alpha-numerical characters, either printed or hand-printed. The syntactical structure of the code is known and follows the regular expression  $L^6 D^2 S [0 - 7] D L D^3 L$  where  $L = [A - Z]$ ,  $D = [0 - 9]$  and  $S = [A, B, C, D, E, H, L, M, P, R, S, T]$ . The last character is a control key which validates the sequence of the 15 first characters.

The main difficulty of the recognition task is that the image quality was very low and the field was very noisy : the lines of the form were degraded, the scanning process distorted the image or the image was faxed, to name a few of the difficulties. As a result, since our usual cleaning procedures could not be used to extract clean characters, we decided to train the character recognizer directly on the noisy images.

We have extracted snippets of characters using the field annotation : each field was split into 16 zones of equal size. We have kept only snippets corresponding to digits. This process has led to a database of 250 257 snippets of digits. We refer to this database as A2iA-AV-Digit. We have randomly split the database in 240 257 images for training and 10 000 images for testing. We also randomly selected 60 000 examples from the train set to define a subset whose size is comparable to MNIST train set. We refer to this database as A2iA-AV-Digit-60k

### 2.2 Compared classifiers

We have compared three classifiers amongst those that achieve the best classification results on MNIST : Support vector machines with RBF kernel[2], k-nearest neighbors with image distortion models[6], and Convolution neural networks[7].

**SVM :** The SVM classifier[2] was trained using the LibSVM<sup>1</sup> software package. We have used a Radial Basis Function kernel given by  $\mathcal{K}(x, x') = \exp(-\gamma \|x - x'\|^2)$ . We have optimized the penalty parameter  $C$  and the radius of the RBF kernel  $\gamma$  by using cross-validation on the train set. The range of values tested were  $C \in [2^1, 2^3, 2^5, 2^7, 2^9, 2^{11}]$  and  $\gamma \in [2^{-3}, 2^{-5}, 2^{-7}, 2^{-9}, 2^{-11}]$  and then refined with smaller steps around the best values. The best values of the hyper-parameters were  $C = 2^8$  and  $\gamma = 2^{-5.2}$  for MNIST,  $C = 2^5$  and  $\gamma = 2^{-5}$  for Extended-MNIST,  $C = 2^5$  and  $\gamma = 2^{-7.5}$  for A2iA-AV-Digit-60k and  $C = 2^5$  and  $\gamma = 2^{-7.5}$  for A2iA-AV-Digit.

**kNN with image distortion models :** Keyzers *et al.*[6] have proposed several image distortion models and applied them to digit recognition and medical image classification. One of them, the Image Distortion Model (IDM), is a computationally simple model yet achieving good results on several machine vision tasks including digit recognition. We have built a classifier based on IDM using the W2D software package provided by C. Gollan<sup>2</sup>. In our experiments, we used the output of horizontal and vertical Sobel filters on a 3\*3 context (18 coefficients) as local features and the Euclidean distance to compare the feature vectors. Once the distance between the test images and all the reference images were computed, we used a k-Nearest Neighbor classifier to predict the class of the test image. We have estimated the best value for  $k \in [1, 20]$  by cross-validation on the train set. It is worth noting that this classifier did not scale well when the size of the training size increased since it required to compute the distance between each test example and all the training examples. This model could not be

<sup>1</sup><http://www.csie.ntu.edu.tw/~cjlin/libsvm/>

<sup>2</sup><http://www-i6.informatik.rwth-aachen.de/~gollan/w2d.html>

trained and tested on the large database Extended-MNIST and A2iA-AV-Digit.

**Convolutional neural network :** This model, based on alternating layers of convolutions and subsampling, has been originally proposed by Le Cun [7] and successfully applied since then to several image recognition tasks amongst which digits recognition [8]. We have modified the architecture of the model proposed by Le Cun in order to replace the Gaussian based classifier by a multi-layer perceptron (MLP). The input of the classifier was the image to be classified and the output is a vector of probabilities associated to each class of the problem. For example, in the case of digit recognition, the size of the output vector is 10. Internally, our model was composed of 6 layers : 5 layers for the extraction of features (3 convolutions and 2 subsampling) and 1 hidden layer with softmax outputs for the MLP classifier.

The classifier was trained iteratively using stochastic back-propagation of the gradient . The cost function used during training was the Kullback-Leibler divergence between the target value and the output of the MLP.

### 2.3 Recognition results

The recognition results of the three classifiers on 4 different databases is presented on Table 1. Our results on MNIST are coherent with those reported on this database with the same models<sup>3</sup>. As expected, the extension of the MNIST training set with elastic distortions allowed to reduce the error rate of both the Convolution Network and the SVM classifier. For computational reasons, the KNN-IDM model could be tested neither on the Extended-MNIST database nor on the A2iA-AV-Digits database.

Overall, the error rates obtained on the A2iA-AV-Digit-60k database are dramatically higher than on MNIST. Two classifiers with less than 1% error rate on MNIST, the Convolutional Network and the kNN-IDM, achieve respectively 2.27% and 11.57% error rate on A2iA-AV-Digit-60. The error rate of the SVM-RBF classifier also dramatically increases from 1.43% to 9.67%. It is important to note that when the amount of training data increases, the error rate decreases as shown by the results on the A2iA-AV-Digit database. The error rate reduction reaches almost 50% for the convolutional network and the SVM classifier.

To conclude, these results show that one can not extrapolate the recognition results obtained on MNIST to a real world digit database with more noise. Classifiers with very similar results on MNIST obtained very different results on our database, the best classifier on MNIST giving the worst result. We think that this result should encourage researchers to be more critical in using MNIST or an

	ConvNet	SVM-RBF	kNN-IDM
MNIST	0.88±0.18	1.43±0.23	0.86±0.18
MNIST extended	0.65±0.16	0.92±0.19	-
A2iA-AV-Digit-60k	2.27±0.29	9.67±0.58	11.57±0.63
A2iA-AV-Digit	1.14±0.21	5.12±0.43	-

**Table 1. Error rate (with binomial confidence intervals at 0.05%) of the three compared classifiers on the test set of four databases.**

equivalent database to evaluate their algorithms. In the next section, we present and evaluate a system based on character recognition dedicated to the recognition of a complete character field.

## 3 Handwritten floating fields recognition

The recognition of isolated digits is never a task by itself in applications but is always one part in a more complex task. In this section, we define a task which is based on character recognition : floating field recognition.

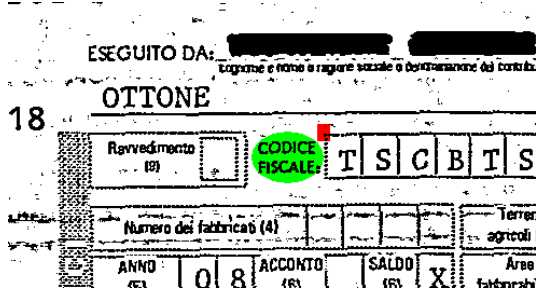
### 3.1 Floating field recognition task

On the database A2iA-AV, the real task is the identification of the sender of the fiscal form. Amongst all the informations available on the form, the main identification number is the fiscal code, which is uniquely associated to each possible sender of the form. The identification of the sender of the form can thus be achieved by recognizing the fiscal code. However, the images show a large variability : position of the form during scanning (the scanning is not normalized), layout of the forms (several types of forms exist), scanning noise and deformation of the image. This variability induces that the location of the fiscal code on the image is not fixed, and that the location of the field is itself a problem, even if one can restrict the area in which the field is expected to be located. We refer to this task as a *floating field recognition* task, which is composed of 2 sub-tasks : the field location and field recognition.

### 3.2 Field location

The goal of this task was to locate the upper left corner of the fiscal code field, shown as a red square on Figure 2. We have restricted the search zone for the fiscal code field to an area of 300 pixels width and 200 pixels height. The detection of the upper-left corner of the field was based on the detection of two keywords : *Codice* and *Fiscale*. The detection of these two keywords was independent and was combined to predict the location of the field. The upper-left

<sup>3</sup><http://yann.lecun.com/exdb/mnist/>



**Figure 2. An example of the upper-left corner of the fiscal code field to be located (red square) and the anchor keywords to be detected (green ellipse).**

corner of the field was predicted in an area of  $16 * 11$  pixels size. In 80.7% of the images in the test set, either one of the two keywords or both keywords were found and the location of the upper-left corner of the field could be predicted. In the case where no keyword was found, either the field was rejected (we refer to this system as *floating field recognition with keyword location*) or a full search of the zone with the recognizer was done (we refer to this system as *floating field recognition with keyword and full search*). We did not directly evaluate the accuracy of the predicted location of the field but rather evaluated the global system (field location and field recognition).

### 3.3 Fixed field recognizer

The field recognizer was based on the Convolution Neural Network character recognizer presented in section 2.2. The character recognizer has been trained on the full database of character snippets extracted from the fiscal code field. The classifier has been trained on 31 classes : from the 36 initial classes (26 letters plus 10 digits), we merged the following classes : 0/O, 1/I, 2/Z, 5/S and 8/B. Confusion between these classes were expected to be solved by the control key on the field. The error rate of the character recognizer on the character test database was 5.4%.

Given a field location as presented in section 3.2, the field zone was divided into 16 snippets of equal width. On each snippet, the character recognizer computed the posterior probability of each character class. The sequence of 16 characters which composes the fiscal code was modeled by a weighted finite state transducer (WFST)[1] denoted  $\mathcal{R}$ . This WFST is defined by a 8-tuple  $(\Sigma, \Delta, Q, I, F, E, \lambda, \rho)$  :

- The input alphabet  $\Sigma$  was the 31 classes of the character recognizer; The output alphabet  $\Delta$  was equal to  $\Sigma$ ;

- The set of states  $Q$  was composed of  $K+1$  states, where  $K$  was the number of characters in the field, in our case  $K = 16$ . The states were numbered from 0 to  $K$ ;
- The set of initial state  $I$  was composed of only one state, corresponding to the first character in the field; The set of final state  $F$  was composed of only one state, the supplementary state which does not correspond to a character;
- The set of transitions  $E \subset Q \times \Sigma \times \Sigma \times Q$  was  $\{\forall q_i \in Q, \forall l \in \Sigma, (q_i, l, l, q_{i+1})\}$ . The weight associated to each transition labeled by  $l$  was  $-\log(P(l))$ , where  $P(l)$  is the probability associated to the character  $l$  by the character recognizer. The WFST was thus defined on the tropical semi-ring  $(\mathbb{R}_+ \cup \infty, \min, +, \infty, 0)$ ;
- The initial output function mapping  $I$  and the final output function mapping  $F$  were null (defined by  $q \rightarrow \epsilon$ ).

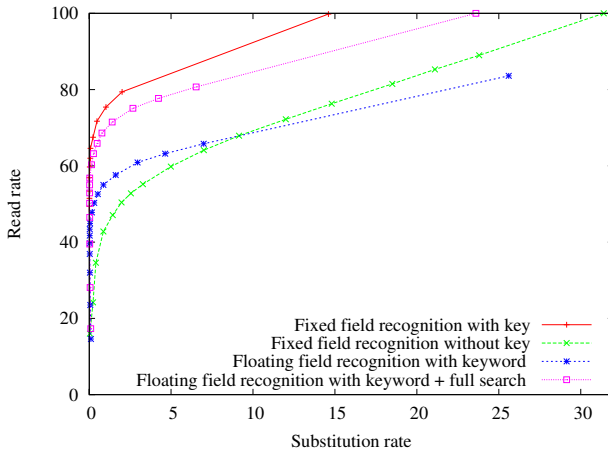
The regular expression defined in section 2.1, which corresponds to the valid forms of fiscal code, was converted into a WFST denoted  $\mathcal{C}$ : each set of characters  $(L, D, S, [0 - 7])$  corresponded to a set of transitions between two states. For each transition, the input and output labels were equal except for the merged classes (0/O, 1/I, 2/Z, 5/S and 8/B) for which 2 transitions were created, one for each possible class in the merge. The cost associated to each transition was identical and null.

The WFST resulting from the composition  $\mathcal{R} \circ \mathcal{C}$  modeled the recognized digit sequences that respect the syntactic constraints of the fiscal code. We have extracted the  $N$  best paths from the composed transducer (in our experiments,  $N=100$ ) and we have filtered the corresponding recognized codes with the control key function. The recognition result of the field recognizer was the filtered recognized code with the lowest cost  $c$ . This cost was transformed into a confidence score with  $\exp(-c)$ .

### 3.4 Field recognition results

We have tested our field recognizer on the A2iA-AV-Field database in two experimental settings : first with the annotated location of the field, referred to as *fixed field recognition* and second with our field locator, referred to as *floating field recognition*. The recognition results are reported using read/substitution curves. Since the field recognizer provided a confidence score associated to its prediction, some of the recognition results can be rejected if the confidence score is below a threshold  $s$ . For a given threshold, the substitution and read rates are given by

$$\text{Read}(s) = \frac{N_a(s)}{N_t} \quad \text{Substitution}(s) = \frac{N_i(s)}{N_a(s)}$$



**Figure 3. Fixed and floating field recognition results : read rate with respect to the substitution rate for four experimental settings.**

where  $N_a(s)$  is the number of recognition results whose score is above the threshold (accepted results),  $N_i(s)$  the number of incorrect recognition results whose score is above the threshold and  $N_t$  the total number of fields to be recognized. By setting the threshold adequately, one can obtain a low substitution rate, which is the key to a successful real world application.

The Fixed field recognition results are shown on Figure 3. The first main result is the importance of the validation key : without a validation key, the field recognition error rate on all the documents is 31.4% whereas with the validation key, the error rate dropped at 14.6%. The second main result is the reliability of the confidence score associated to the recognition results : by reducing the read rate to 65%, an error rate as low as 0.01% can be achieved.

The floating field recognition results are also shown on Figure 3. As expected, the automatic location of the field has an impact on the recognition results : at a substitution rate of 15% , the read rate is 100% for fixed field recognition, and only 73% for *Floating field recognition with keyword*. In the case where no keyword is found for the location of the field, the recognizer is used for a full search of the zone : the recognition results are presented on Figure 3 as *Floating field recognition with keyword + full search*. Even if this method was computationally intensive, it yielded very good results : at low substitution rate (<1%), the read rate is only 5% lower than with the annotation of the location of the field. With this fully automatic system (location and recognition), an error rate at 0.01% can be achieved with a read rate at 60%, which is suitable for a real world application.

## 4 Conclusion

In this paper, we have revisited two aspects of character recognition. First we think that current databases of isolated characters are too simple and have been used for too long in the community to reflect the state-of-the-art of character recognition for industrial applications. We have tested three state-of-the-art classifiers on MNIST and on a database extracted from a real-world application and we have shown that the error rate increases dramatically from around 1% on MNIST and up to 10% on our base. This shows that the three classifiers are not equivalent with respect to their resistance to noise. Second, we have described and evaluated a system based on isolated character recognition and designed for the recognition of floating character fields. We have shown that several aspects, other than the recognition rate on isolated characters, are important for an industrial application : the localization of the field, its syntactical structure, the presence of a control key and, overall, the confidence score associated to the recognition result, which permits to reject the unreliable results and thus to lower the error rate to a level acceptable by the final client.

**Acknowledgment:** the authors wish to thank Christian Gollan from the Computer Science Department of RWTH Aachen University for his help with the W2D software.

## References

- [1] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri. Openfst: a general and efficient weighted finite-state transducer library. In *Proceedings of the Conference on Implementation and Application of Automata*, pages 11–23, 2007.
- [2] C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.
- [3] N. Gorski. *Digital Document Processing*, chapter Bank Cheque Data Mining: Integrated Cheque Recognition Technologies, pages 437–458. 2007.
- [4] E. Grosicki, M. Carré, J.-M. Brodin, and E. Geoffrois. Results of the second rimes evaluation campaign for handwritten mail processing. *Proceedings of the International Conference on Document Analysis and Recognition*, 2009.
- [5] J. Hull. A database for handwritten text recognition research. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(5):550–554, 1994.
- [6] D. Keysers, T. Deselaers, C. Gollan, and H. Ney. Deformation models for image recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(8):1422–1435, 2007.
- [7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [8] P. Simard, D. Steinkraus, and J. Platt. Best practice for convolutional neural networks applied to visual document analysis. In *International Conference on Document Analysis and Recognition*, pages 958–962, 2003.