SIMPLE FEATURE EXTRACTION FOR HANDWRITTEN CHARACTER RECOGNITION

P. Pedrazzi, A.M. Colla

Elsag Bailey - Finmeccanica S.p.A.
Via G. Puccini, 2 - 16154 Genova (ITALY)
tel. +39-10-658-2767 fax +39-10-658-2694
e-mail : pedrazzi@elsag.it

ABSTRACT

This paper deals with a simple and effective set of features for (hanprinted) character representation in automatic reading systems. These features, computed within regularly placed windows spanning the character bitmap, consist of a combination of average pixel density and measures of local alignment along some directions. Patterns from different databases can be accommodated by choosing a variable window size.

These features used in conjunction with a neural classifier (MLP) yielded a very high accuracy on several handprinted character databases, including NIST's ones. Moreover they are easily implementable in VLSI, with throughputs as high as 250,000 characters/sec.

1. INTRODUCTION

The high variability usually found in handwritten text has forced many researchers in the field of OCR to find more and more powerful techniques to tackle the problem. Among all the different modules a reading system may be decomposed in, the *feature extraction module* has always received great attention. In fact, while clustering and classification tasks are quite general topics - and several good techniques do exist - there still is huge uncertainty about which kind of *features* will best perform in the relatively recent topic of handwritten OCR. Proposed methods span from *structural feature primitives* [1] to *moment invariants* [2], *stroke analysis* [3], use of *morphological operators* [4] or other operators and mathematical transforms [5]. Indeed, also Neural Networks, fed with raw data, have been used to *learn* meaningful features [6]. Finally, in order to improve discrimination, some authors put together sets of features of different kinds, trying to overcome typical drawbacks of each single set [7].

Moreover, some classifiers, like Neural Net Classifiers, impose that the same number (and kind) of features has to be produced for every sample; this is not true in the case of most structural feature extractors.

On the other hand, practical applications are often demanding as regards throughput, and this limits the accettable complexity of the feature extractor, as well as of any other module in a reading system. Some feature extractors make use of such sub-functions as *contour extraction* or *skeletonization*, which are rather time-consuming algorithms, besides being hazardous when applied to damaged, incomplete, or noisy bitmaps.

In this paper we present a novel set of features which are simple to calculate, fixed in number, robust enough with respect to noise, and which have proved highly discriminative both on our proprietary databases and on the well-known *NIST Test Data 1 (TD1)* [8].

In next section we introduce the feature set. In the third section we describe the experimental setup, in which the feature set is used in conjunction with a classical MLP classifier. In the fourth section we illustrate and discuss the results on NIST *TD1*. In the last section we draw some conclusions.

2. THE FEATURE SET

Our method is a development of a previous feature extractor based on *gray pixels* [9]. The module is fed with the bitmap of a single character, previously deskewed and normalized. Our current implementation works with binary pixels, but it could be easily modified to accept gray level images as well.

First, the input image is partitioned into subimages (*windows*). The windows are usually placed on a regular grid and are partially overlapped. A fixed set of operators is applied to each window. The feature set consists of the outputs generated one by each operator in each window.

The first operator is a simple bit counter that calculates the *average pixel density* in the window, as in the previous simpler feature extractor. The other operators

0-8186-7310-9/95 $4.00

try to estimate at which extent the black (meaningful) pixels in the window are aligned along some directions. For each direction of interest, we define a set of N, equally spaced, straight lines, that span the whole window and that are – at least approximatively – parallel to the chosen direction. Along each line $j \in \{1, N\}$ the number n_j of black pixels is computed. Then we apply to n_j a non linear function that emphasizes the presence of the black pixels. Among all the possible choices, we found that the following quasi-exponential function is effective:

$$f(n) = \begin{cases} 0 & \text{if } n < 2 \\ 2^{(n-2)} & \text{if } n \geq 2 \end{cases}$$

Summing up all the contributions $f(n_j)$ from the N lines, we obtain the so-called *primitive feature* along a given direction.

This simple scheme can be enriched in many ways. For example, the analysis of some directions could require a double set of lines; in this case the primitive features must be combined. Moreover some *affine* combination of the primitive features can be used.

In the reported experiments we used a feature vector of 140 elements generated from 35 windows placed on a regular 7 × 5 grid.

3. EXPERIMENTAL SETUP

The feature set was tested on several databases in conjunction with a classical MLP classifier. The databases consist of NIST images and Elsag images. They were all prepared according to our Company's usual preprocessing scheme, that includes deskewing and normalization of each character to a fixed size bitmap. After pre-processing, the only difference between Elsag data and NIST data was the bitmap size: 22 × 30 pixels for Elsag's databases, 24 × 36 for NIST's databases.

The classifier was a one-hidden layer MLP, trained with standard on-line Backpropagation [10]. As activation function we used the *logistic function*. The output layer contained one node for each class, and classification was accomplished by a simple *maximum response* strategy. Rejection was based on a single threshold applied to the difference between the two highest output values.

It should be noted that the experiments were run according to the same methodology adopted in the well known 1992 Contest [8]. We prepared two disjointed sets of data, the *training set* and the *cross validation set* [11]. We trained the networks with the former, and we tested them on the latter. According to Vapnik [12], using cross validation sets allows to estimate the prediction risk, that is the expected performance in classification of unknown data. Therefore, for each experiment, the *best* network was identified as the best performing network on the cross validation set, and just that network was used in the subsequent test phase on the database TD1. The results are reported in next section.

Two different kinds of experiments were run. In the former (indexed by 1), we built the training set and the cross-validation set from the NIST *Special Database 3 (SD3)* only. In the latter (indexed by 2), we generated a composite training set by mixing SD3 with one of Elsag's proprietary databases *(DBP)*. Because of the difference in size, we used 6 × 6 windows for NIST data and 5 × 5 windows for DBP data, in order to get in both cases the same number of analysis windows, hence the same number of features. Both training and cross-validation sets were well balanced. The test set TD1 was processed in exactly the same way as SD3.

4. EXPERIMENTAL RESULTS

Experiments were carried out with a variety of parameterizations; each experiment was run 3 times with different random network initializations. Here are reported the results obtained with the best performing nets. Details about the experiments can be found in Tables 1 and 2. We couldn't experiment any composite training set in the case of lowercase letters because there was no suitable proprietary database available.

Exp.	Training Set		C-V Set		Test
	SD3	*DBP*	*SD3*	*DBP*	*TD1*
Dig_1	60,000	–	60,000	–	58,646
Dig_2	40,000	20,000	40,000	20,000	58,646
Upp_1	29,969	–	14,968	–	11,941
Upp_2	29,969	32,173	14,968	15,002	11,941
Low	30,216	–	15,097	–	12,000

Table 1: *Description of experiments: number of patterns in each data set.*

In Table 3 we report, for each experiment, the average error rate (misclassifications) on three runs at zero rejection rate.

In Table 4 the error rates obtained with the best performing networks are listed at given rejection points. The *error rate* is defined as the fraction of the accepted (i.e. not rejected) patterns that are misclassified, that is, as the complement to accuracy. The values reported in Table 4 are also graphicated in Figure 1.
Comparing these results with those reported at NIST

Exp.	N_{hid}	N_{out}
Dig_1	185	10
Dig_2	185	10
Upp_1	185	26
Upp_2	280	26
Low	280	26

Table 2: *Description of experiments: number of MLP nodes. The input layer has 140 nodes.*

	Digits		Upperc.		Low.
	Dig_1	Dig_2	Upp_1	Upp_2	Low
Best Res.	(w) ATT_1 3.16	(*) AEG 2.90	(w) AEG 3.74	—	(w) AEG 12.74
Error	2.97	2.55	4.71	4.02	12.72

Table 3: *Performance on NIST Test Data 1: average error rate on 3 runs vs. either* (w) *1992 NIST Contest winners or* (*) *best subsequently reported results.*

First Census Conference [8], it can be easily seen that our results are at the top in the case of digit and lowercase recognition. In the case of uppercase, AEG result is still slightly superior.

In order to highlight the true contribution of our feature set to the overall accuracy results, in Figure 1 the accuracy/rejection curve obtained with the simpler "gray pixels" character representation, Upp_{1G}, is also represented. The relevant experimental setup is as follows: data sets as in Upp_1, 500 hidden nodes (in order to obtain nets with about the same number of weights as the ones fed with the complete feature set), character representation consisting of only the first primitive feature, computed on the same windows as above. This curve shows a constantly worse performance than the curve relevant to Upp_1, where the whole feature set has been employed.

The time required to calculate the feature set for one character ranges between 0.6 ms (PC 486, 66 MHz) and 0.1 ms (DEC Alpha, 200 MHz). A much higher throughput can be obtained implementing the feature extractor in an analog CMOS VLSI chip. A preliminary study on a simplified version of our feature set shows that computing times as low as 4 μs per pattern can easily be obtained [13].

Rej. Rate	Digits		Upperc.		Lowerc.
	Dig_1	Dig_2	Upp_1	Upp_2	Low
0%	2.86	2.46	4.56	3.92	12.60
2%	1.94	1.56	3.52	2.97	11.71
5%	1.09	0.82	2.35	1.99	10.35
10%	0.47	0.38	1.32	1.12	8.49
15%	0.27	0.22	0.85	0.74	6.84
20%	0.17	0.15	0.66	0.58	5.52

Table 4: *Performance on NIST Test Data 1: best out of 3 runs.*

5. CONCLUSIONS

We presented a novel feature extraction method that couples good performances with intrinsic simplicity.

Performances with our method achieve a prominent position when compared on a world wide known test bed. These scores are particulary valuable taking into account the fact that the recognition phase was carried out by a single, quite general neural network. The ability to feed MLP nets and the straightforwardness of the feature extraction method allow to build up an accurate and fast recognizer. The possibility of an easy implementation of this feature extractor in a VLSI chip is even more promising, since VLSI implementations of the MLP classifier (forward phase) are now available.

Finally, it can be noted how simple is to deal with data with different size normalizations, just by scaling the size of the analysis window.

6. REFERENCES

[1] A.Y. Commike and J.J. Hull, *Rule Learning for Syntactic Pattern Recognition*, USPS Fourth Adv Tech Conf, Nov 1990, pp. 621–633 (1990)

[2] S.O. Belkasim, M. Shridhar and M. Ahmadi, *Pattern Recognition with Moments Invariants: A Comparative Study And New Results*, Pattern Recognition, **24**(12), pp. 1117–1138 (1991)

[3] Z. Zhang, I. Hartmann, J. Guo & R. Suchenwirth, *A Recognition Method of Printed Chinese Characters by Feature Combination*, Int. Journal of Research & Engineering – Postal Applications, inaugural issue, pp. 77-82 (1989)

[4] T. Kanungo and R.M. Haralick, *Character Recognition Using Mathematical Morphology*, USPS Fourth Adv Tech Conf, Nov 1990, pp. 973–986 (1990)

[5] P.J. Grother, *Karhunen Loève Feature Extraction For Neural Handwritten Character Recognition*, Technical Report NISTIR 4824 (1992)

[6] Y. LeCun et alii, *Backpropagation Applied to Handwritten Zip Code Recognition*, Neural Computation, **1**, pp. 541–551 (1989)

[7] L. Heutte et alii, *Handwritten Numeral Recognition Based on Multiple Feature Extractors*, Proceedings of 2^{nd} ICDAR, Tsukuba, Japan, pp. 167–170 (1993)

[8] *The First Census Optical Character Recognition System Conference*, Technical Report NISTIR 4912 (1992)

[9] R. Battiti and A.M. Colla, *A Parallel Implementation of Neural Models for Pattern Recognition*, in E.R. Caianiello (Ed.), *Forth Italian Workshop PARALLEL ARCHITECTURES AND NEURAL NETWORKS* (Vietri Sul Mare, Italy, May 8-10 1991), pp. 115–122, World Scientific Publishing Co., Singapore (1991)

[10] J. Hertz, A. Krogh, R.G. Palmer, *AN INTRODUCTION TO NEURAL COMPUTATION*, Addison-Wesley (1991)

[11] M. Stone, *Cross-validatory choice and assessment of statistical predictions*, Roy. Stat. Soc. **B36** (1974)

[12] V. Vapnik, *ESTIMATION OF DEPENDENCES BASED ON EMPIRICAL DATA*, Springer-Verlag (1982)

[13] D.D. Caviglia, M. Valle, A.M. Rossi, M. Vincentelli, G.M. Bo, P.P. Colangelo, P. Pedrazzi, A.M. Colla, *Feature Extraction Circuit for Optical Character Recognition*, Electronic Letters, **30** (10), pp. 769–770 (May 1994)

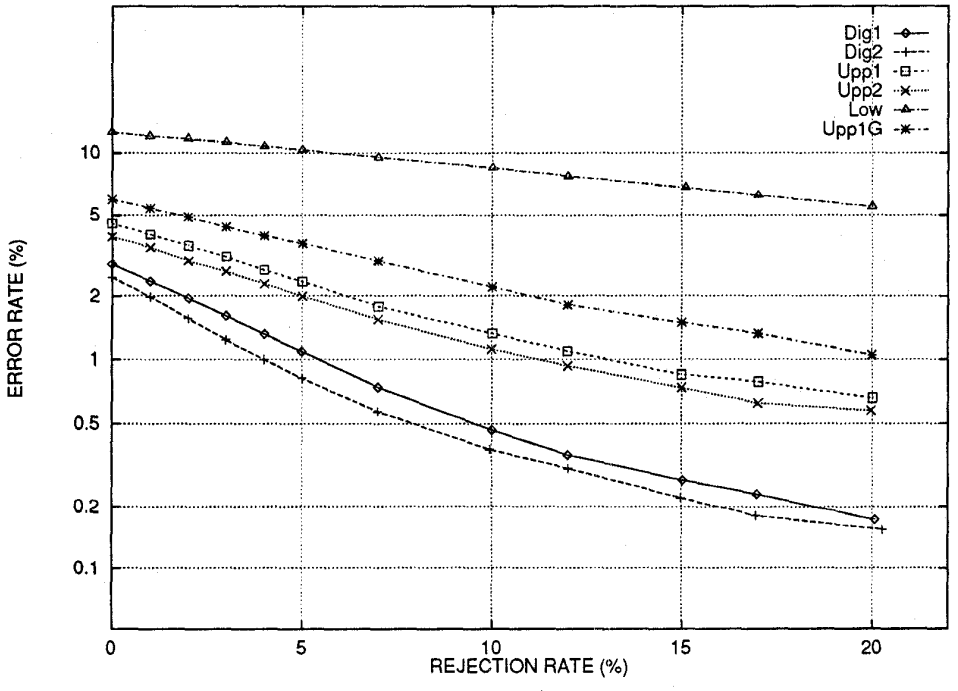

Figure 1: *Accuracy/rejection curves on database TD1.*