A NEURAL PREDICTIVE APPROACH FOR ON-LINE CURSIVE SCRIPT RECOGNITION

S. Garcia-Salicetti*', P. Gallinari**, B. Dorizzi*, A. Mellouk**, D. Fanchon***

*Institut National des Télécommunications, Dept. SIM
9 rue Charles Fourier, 91011 Evry, France
**LAFORIA-IBP UA CNRS 1095
Tour 46-00 Boite 169, Université Paris VI
4 Place Jussieu, 75252 Paris Cedex 05
France

ABSTRACT

We present a neural prediction system for on-line writer-independent character recognition as a first step towards a word recognition system. The input feature vectors contain the pen trajectory information, recorded by a digitizing tablet. Each letter is modeled by a variable number of predictive Neural Networks, depending on its length. Successive parts of a letter are modeled by different Multilayer Neural Networks, only transitions from each one to itself or to its right neighbors being permitted. To deal with the great variability of cursive handwriting, we introduce a holistic approach for both Learning and Recognition, combining Neural Networks and Dynamic Programming techniques. Our system is able to recognize strongly distorted and truncated letters, obtained by automatic segmentation of 10000 words from 10 different writers. Even on such databases, unappropriate to character recognition (letters in it were not recorded as handwritten isolated characters), quite good recognition rates are obtained.

1. INTRODUCTION

Most systems for word recognition involve first segmentation of words into letters followed by recognition of the resulting characters. In other words, there is in such systems no interaction between segmentation and recognition [1,2]. The major problem they encounter is indeed the search for a robust *a priori* segmentation of words into letters. They are certainly very efficient in the mono-scriptor framework and can offer good performances in the multi-scriptor framework with a limited number of writers, because of limited handwriting variability. But their structural static segmentation does not allow them to be performing in the omni-scriptor context because of the tremendous handwriting variability of the latter.

Some systems try to overcome these limitations; they rely on the theory of Hidden Markov Models (HMMs) in order to tune segmentation to letter samples [3]. Our approach is related to HMMs in a statistical way; it performs adaptive segmentation due to its ability to take into account variability of time spent in each state. Actually, this property makes our approach specifically adapted to the omni-scriptor framework. It has already given good performances when used for Speech Recognition [4,5]. Moreover, it is easy to implement because of a much simpler parameter reestimation procedure than the one proper to the classical HMM framework. The use of an approximate Maximum Likelihood criterion, that we will further detail, permits such a simplification.

Furthermore, our use of neural emission models has the virtue of allowing non-linear prediction on each feature vector extracted, called "frame", from a given context of frames. Also, such a context is flexible in our model since its definition is totally independent of feature extraction. Indeed, a change in context dimensions only implies a change on networks' input layers' size. Low-level contextual information is thus available in an explicit and simple way. Neural models enrich this way the classical use of HMMs, in which context is fixed in the model once feature extraction has been performed.

2. SYSTEM OVERVIEW

2.1 Databases

The databases used in this work are letter databases obtained from 10 000 handwritten words from 10 writers, that were split into "letters" by an automatic segmentation procedure [6]. That means that our databases contain distorted characters as well as truncated ones, depending of *where* in the word's trajectories, segmentation was automatically operated. *Figure 5* (at the end) shows some examples of characters from our databases. We are obviously quite far from the kind of database used in other character recognition studies, where characters have been drawn separately [7]. Notice that handwriting variability is very important in our databases, as *Figure 5* also shows.

Unfortunately, there exists disparities between the number of examples of different letters in our databases, forcing us to discard a few letter-models, like 'i', 'z' and 'v', for which we have too few examples to train the corresponding letter-.models. Our future aim is of course to be able to test our system on a larger reference database, more precisely on the UNIPEN Project databases [8].

0-7803-2431-5/95 $4.00

2.2 Feature Extraction

The data collection of each letter was obtained from words written with a fixed scale on a digitizing tablet that samples the pen trajectory at the frequency of 200 points per second. Data finally appear as a sequence of (x,y) coordinates. Each feature vector (frame) is extracted from a fixed number of points in a trajectory, a "window", that shifts along the trajectory by a certain amount of points, this way permitting an overlap. We thus use a space displacement approach for parameter extraction. It has the advantage of being flexible, since we can change the size of the considered "window" as well as the space displacement step. As shown in *Figure 1*, extracted parameters per "window" are the oriented angle α, and the algebraic area between vector AB (A and B being respectively the first and last point of the "window") and the curve portion contained in the sliding window, as well as the percentage of pen-lifts encountered [9]. This feature extraction procedure is neither too local and thus not too sensitive to very little variations of the pen's trajectory (it therefore gives similar parameter representation to very close curve shapes). It is nor too global and, thus, not likely to extract uninforming parameters for the networks in a discriminative point of view. This compromise is thus robust enough for our neural predictive approach.

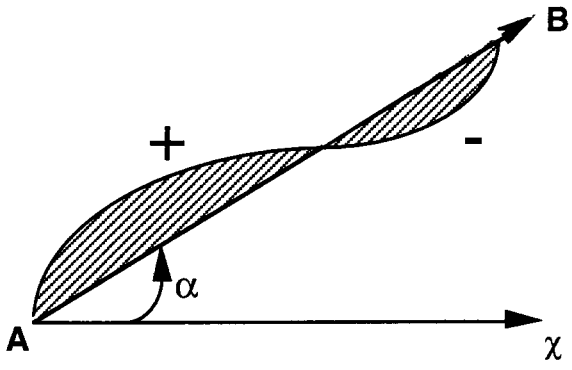

Fig.1: the feature extraction step

2.3 The Neural Prediction System

Each letter is modeled as being emitted by a given number of multilayer networks (states). This number depends on the average length of letters. "Long" letters are modeled as being generated by 5 networks, "short" ones as being emitted by 4 networks. Each network is called a "state-model". Transitions between state-models are ruled by the topology at hand; we have considered a "Left-to-Right" topology with an extra restriction, namely the allowance of transitions from each state only to itself or to its right neighbor (see *Figure 2*).

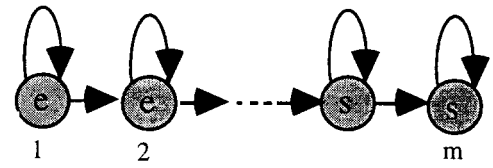

Fig.2: the simple left-to-right topology considered for m states
e: entry state; s: exit state

Networks' output is a non-linear prediction of a given frame starting from its predefined context, a predictive context of three frames (see *Figure 3*).

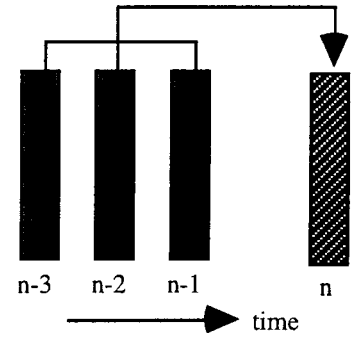

Fig.3: the predictive context of three frames

During training, prediction errors are processed through Dynamic Programming. Shortest path segmentation is performed before adjusting parameters to the current letter trajectory presented (see *Figure 4*).

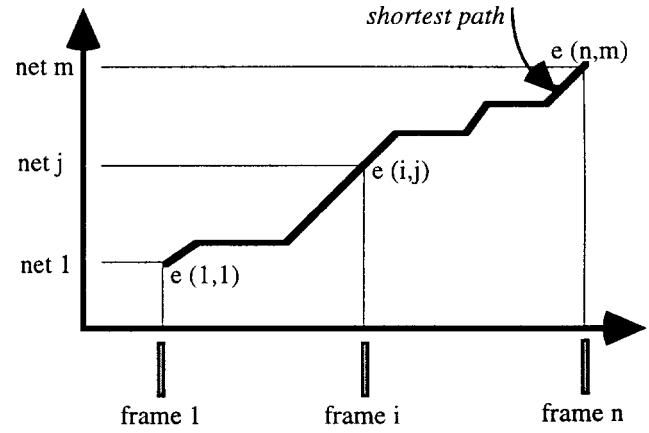

Fig.4: shortest path segmentation on the prediction error matrix M= (e (i,j))

The emission models' parameters are reestimated stochastically by back-propagation of the corresponding global error. Cross-validation is performed to stop the learning process. During recognition, each letter to be recognized is presented to all letter-models. The winning model is the model offering the best score relative to the global prediction error.

3. STATISTICAL INTERPRETATION

The use of a purely predictive context of 3 frames allows us to consider each state-model as a non-linear auto-regressive emission model of order 3. Its corresponding trajectory, which varies from one example of a letter to another, according to shortest path segmentation, is therefore governed by *fixed* dynamics. On the other hand, the letter-model as a whole is a non-linear auto-regressive process driven by a noise source, with *variable* dynamics. This variability is indeed governed by the topology chosen for the letter-model, as we will explain below.
The letter emission model is then:

$$O_t = F_{\omega(q_t)}(O_{T(t)}) + \varepsilon_t$$

where $O_{T(t)}$ is the prediction context for O_t, $F_{\omega(q_t)}$ the non-linear function computed by the emission model of parameters $\omega(q_t)$, q_t being the state occupied at time t, and ε_t a gaussian noise whose realizations are supposed to be independent. We consider an approximate Maximum Likelihood criterion for the model parameters' learning process [10]:

$$\max_\lambda [\, \max_q [\, P(O, q / \lambda)\,]\,] \ .$$

λ being the N-states Hidden Markov Model model of a given letter, and q denoting a state sequence. Optimization is thus performed alternatively on networks' parameters and on segmentation. The first step of the whole reestimation procedure, the search of the optimal state sequence is:

$$\arg\max_q [\, P(O, q/\lambda)] = \arg\max_q [P(O / q, \lambda) \,.\, P(q/\lambda)] \ ;$$

but, if we consider all transition probabilities equal in the case of an ergodic model, we have:

$$P(q/\lambda) = P(q'/\lambda) \quad \forall q \neq q' \ ;$$

and, therefore:

$$\max_q [\, P(O, q/\lambda)\,] = \max_q [\, P(O/q, \lambda)\,] \ .$$

Thus, the model's equation leads to:

$$P(O/q, \lambda) = P((\varepsilon_t)_{t=1}^{T} / \lambda)$$

$$= \prod_{t=1}^{T} P(\varepsilon_t = [O_t - F_{\omega(q_t)}(O_{T(t)})] / \lambda)$$

and, because of the gaussian hypothesis, we have:

$$\arg\max_q [\, P(O/q, \lambda)] = \arg\min_q [\sum_{t=1}^{T} \| O_t - F_{\omega(q_t)}(O_{T(t)}) \|^2]$$

The simple Left-to-Right topology we considered is defined by

$$\forall i \quad a_{i,i} + a_{i,i+1} = 1$$

Considering as our cost function the summation of prediction errors per frame corresponds to an approximation of the HMM framework, since an emission probabilities product is maximized instead of the classical term involving also transition probabilities. However, our approach keeps the essential idea of the HMM framework while also enriching the HMM approach with the tools of neural prediction. Therefore, its implementation not only gives good results, but also simplifies the model parameters' reestimation procedure, since the only parameters involved are networks' weights, which are easily reestimated by backpropagation.

4. RESULTS

On databases described in section 2.1, the recognition rate is in average 70% on letters from 'a' to 'y' (we did not take into account 'i', 'v' and 'z' because of lack of examples). More precisely, recognition rates vary from about 40% for letters for which we have few examples (about 100 examples) to more than 90% for letters well represented in our databases (above 1000 examples), even reaching 100%. We consider this result very encouraging since our letter databases are not made for letter recognition (because letters in it were not recorded as handwritten isolated characters but were produced by automatic segmentation), and since we could only test our system on characters that are truncated or elongated or, worse, characters that have *identical shapes while coming from databases corresponding to different letters* (see *Figure 5*). Therefore, letter-models cannot be trained to learn discriminant informations of each letter, and this increases remarkably confusions between different letters. The most important confusions obtained (of at least 20%, and even reaching 40%) are explainable by this phenomenon. The nature of our databases therefore confirm that our system has a great ability to deal with cursive handwriting variability. We hope to improve these results with the use of larger "clean" databases (like the UNIPEN Project's ones). However, the real interest of our approach

should appear at the word level, where our holistic approach for segmentation as well as its interaction with learning and recognition, shall be robust to the variability of links encountered between successive letters in a word. But, at the letter level, it would be better to train letter-models on examples containing discriminant informations of each letter, in order to learn most variability at the word level.
On the other hand, our purpose is to develop the markovian aspect of our letter-models by the introduction of transition probabilities between neural models and the use of durational modeling. This should also improve recognition rates.

REFERENCES

[1] Schomaker L.R.B., Teulings H.L.: "Stroke versus character-based recognition of on-line connected cursive script", From Pixels to Features III: Frontiers in Handwriting Recognition, S. Impedovo and J.C. Simon (eds), 1992, Elsevier Science Publishers B.V.
[2] Tappert C.C., Suen C.Y. and Wakahara T.: "The state of art in Handwriting Recognition", IEEE Transactons on Pattern Analysis and Machine Intelligence", Vol. 12, N°8, August 1990.
[3] Bercu S., Lorette G.: "On-line handwritten word recognition: an approach based on Hidden Markov Models", International Workshop on Frontiers in Handwriting Recognition III, Buffalo, 1993.
[4] Mellouk A., Gallinari P.: "Continous speech recognition by neural spectrum prediction systems", WCNN 93.
[5] Mellouk A., Gallinari P.: "A discriminative neural prediction system for speech recognition", ICASSP 93, II, pp. 533-536, 1993.
[6] Duneau L., Dorizzi B.: "Automatic letter segmentation of labelled cursive words by generation of allographs", ICOHD 93, pp. 210-212, Paris,1993.
[7] Manke S., Bodenhausen U.: "A connectionist recognizer for on-line cursive handwriting recognition", ICASSP 94, II, pp. 633-636.
[8] Guyon I., Schomaker L., Plamondon R., Liberman M., Janet S.: "UNIPEN project of on-line data exchange and recognizer benchmarks", ICPR 94, pp. 29-33, Jerusalem, October 1994.
[9] Duneau L., Dorizzi B.: "On-line cursive script recognition: a system that adapts to an unknown user", ICPR 94, pp. 24-28, Jerusalem, October 1994.
[10] Levin E.: "Hidden control neural architecture modeling of nonlinear time varying systems and its applications", IEEE Transactions on Neural Networks, Vol. 4, N° 1, Jan. 1993.

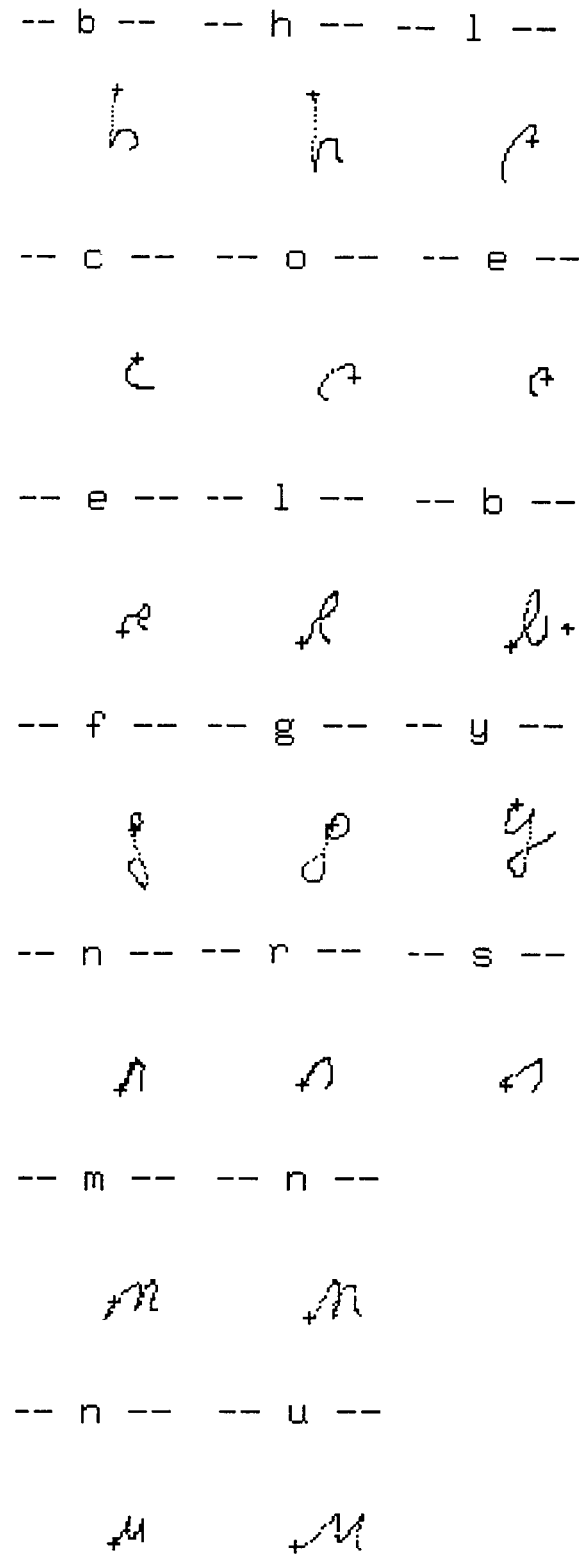

Fig. 5: letter examples truncated, elongated, or with identical shapes