A Hybrid Large Vocabulary Handwritten Word Recognition System using Neural Networks with Hidden Markov Models

Alessandro L. Koerich
Lab. d'Imagerie, de Vision et d'Intelligence Artificielle
École de Technologie Supérieure
Montréal, H3C 1K3, Canada
alekoe@computer.org

Yann Leydier
Lab. Reconnaissance de Formes et Vision
I.N.S.A. de Lyon
Villeurbane, France
yleydier@rfv.insa-lyon.fr

Robert Sabourin
Lab. d'Imagerie, de Vision et d'Intelligence Artificielle
École de Technologie Supérieure
Montréal, H3C 1K3, Canada
sabourin@gpa.etsmtl.ca

Ching Y. Suen
Centre for Pattern Recognition and Machine Intelligence
Concordia University
Montréal, H3G 1M8, Canada
suen@cenparmi.concordia.ca

Abstract

In this paper we present a hybrid recognition system that integrates hidden Markov models (HMM) with neural networks (NN) in a probabilistic framework. The input data is processed first by a lexicon–driven word recognizer based on HMMs to generate a list of the candidate N–best–scoring word hypotheses as well as the segmentation of such word hypotheses into characters. An NN classifier is used to generate a score for each segmented character and in the end, the scores from the HMM and the NN classifiers are combined to optimize performance. Experimental results show that for an 80,000–word vocabulary, the hybrid HMM/NN system improves by about 10% the word recognition rate over the HMM system alone.

1 Introduction

During the last few years, HMMs have become a very popular approach in handwriting recognition. One of the reasons is their higher performance in medium to large vocabulary applications where segmentation–recognition methods are used to cope with the difficulties of segmenting words into characters. Segmentation–recognition methods first loosely segment (oversegment) words into graphemes that ideally consist of either characters or parts of characters, and use dynamic programming techniques together with a lexicon to find the definitive segmentation as well as the best word hypotheses [9, 2, 4]. Many systems use HMMs to model sub–word units (characters) and the Viterbi algorithm to find the best match between a sequence of observations and the models [2, 4, 7]. The Viterbi algorithm is optimal in the sense of maximum likelihood and it looks at the match of the whole sequence of features (observations) before deciding on the most likely state sequence. This is particularly valuable in applications such as handwritten word recognition where an intermediate character may be garbled or lost, but the overall sense of the word may be detectable. On the other hand, the local information is somewhat overlooked. Furthermore, the conditional–independence imposed by the Markov Model (each observation is independent of its neighbors) prevents an HMM from taking full advantage of the correlation that exists among the observations of a single character [12].

Neural network classifiers exhibit powerful discriminative properties and they have been used in handwriting recognition particularly with digits, isolated characters, and words in small vocabularies [11]. However, the use of NNs in the recognition of handwritten words from larger vocabularies depends heavily on a very efficient segmentation scheme. Due to the lack of such an efficient segmentation scheme, NNs are usually employed in combination with other classifiers, e.g. hybrid NN/HMM approaches that use NNs to estimate *a priori* probabilities [9], or that use NNs to validate grapheme hypotheses generated by HMM classifiers [3].

In this paper we propose an approach to integrate NNs and HMMs in a probabilistic framework that takes advantage of the good properties of both methods: the generation

of an N–best list of word hypotheses by the HMM classifier together with the segmentation of each hypothesis into characters and the character modeling properties of the NN classifier. The NN classifier uses the segmentation information provided by the HMM classifier to go back to the input image and extract new features more suitable for isolated character recognition. The NN classifier scores the segments of each N–best word hypothesis and such scores are further combined with the scores generated by the HMM classifier. Finally, the N–best list is reordered according to the new composite scores, shifting up the correct word hypothesis.

2 Word Recognition

The basic problem in handwriting recognition is given an input pattern represented by a sequence of observations $O = (o_1 o_2 \dots o_T)$ where T is the number of observations in the sequence, and a recognition vocabulary represented by R_V corresponding to V unique words, find the word $w \in R_V$ that best matches to the input pattern. The standard approach is to assume a simple probabilistic model of handwriting production whereby a specified word, w, produces an observation sequence O with probability $P(w, O)$. The goal is then to decode the word, based on the observation sequence, so that the decoded word has the maximum *a posteriori* (MAP) probability, i.e.:

$$\hat{w} \ni P(\hat{w}|O) = \max_{w \in R_v} P(w|O) \tag{1}$$

Using Bayes' Rule and assuming that $P(O)$ does not depend on w, and equal *a priori* probabilities of words $P(w)$, the MAP decoding rule can be approximated by:

$$\hat{w} \approx \arg \max_{w \in R_v} P(O|w) \tag{2}$$

The way we compute $P(O|w)$ for large vocabularies is to build statistical models for sub–word units (characters) in an HMM framework, build up word models from these sub–word models using a lexicon to describe the composition of words, and then evaluate the model probabilities via standard concatenation methods and DP–based methods such as the Viterbi algorithm. This procedure is used to decode each word in the lexicon.

The output of the word recognizer is an N–best word hypothesis list ordered according to the *a posteriori* probability assigned to each hypothesis. Furthermore, the information of the segmentation of the word hypotheses into characters is also available and it is obtained by the backtracking of the best state sequence. So, the output of the word recognizer denoted as Ψ_{HMM} can be represented by a triple:

$$\Psi_{\text{HMM}} = \{\hat{L}_n, \hat{S}_n, \hat{P}_n\} \tag{3}$$

where $\hat{L}_n = c_1^n c_2^n \dots c_H^n$ is the label for the n–*th* word hypothesis, and it is given as a sequence of H characters where c_h denotes the h–*th* character within the word; $\hat{S}_n = x_1^n x_2^n \dots x_H^n$ is a set of segments that corresponds to the segmentation of the word into characters in which x_h is the segment that corresponds to the h–th word character; $\hat{P}_n$ is an estimate of the *a posteriori* probability assigned to the n–*th* word hypothesis by the HMM classifier.

3 Isolated Character Recognition

Given the triple $\{\hat{L}_n, \hat{S}_n, \hat{P}_n\}$ provided by the HMM classifier, character alternatives can be located within the word hypotheses by using the segmentation $\hat{S}_n$. Character recognition is introduced to improve the disambiguation among the resulting word hypotheses. Another feature extraction module is used to extract new features from such segments, and a different feature vector is formed. The task of the isolated character classifier is to assign Bayesian *a posteriori* probabilities to such a new feature vector that represents a character given that its class is already known. Further, the probabilities of the individual characters can be combined to generate a new score to the word hypotheses. These new word scores can be combined with the output of the HMM classifier by a suitable rule to build up a hybrid recognition system. Figure 1 shows the integration of the HMM–based word recognition and the isolated character classifier building up a hybrid recognition system.

The hypothesis underlying the hybrid system is that the segmentation of words into characters carried out by the HMM classifier is somewhat reliable. Furthermore, if we look at the N–best list, the difference in the accuracy when considering only the best word hypothesis (TOP1) and the ten best word hypotheses (TOP10) is more than 15% (for an 80,000–word lexicon). This is due to the presence of very similar words in the lexicon that may differ only by one or two characters. So, recognition at the character level is an attempt to better discriminate characters, since such an aspect is somewhat overlooked at the word recognition based on HMMs.

We can rely on many different methods to recognize isolated characters. Handwritten character recognition has been the subject of much attention in the field of handwriting recognition. Several proposals to solve this problem have been presented [5, 10, 1]. However, our problem is slightly different, since we do not want to really perform character classification, that is, assign a class to a given feature vector. In fact, the character class is already known, and we only want to estimate the probability of such a class to represent the feature vector. So, the task of the character classifier is to assign an *a posteriori* probability to each

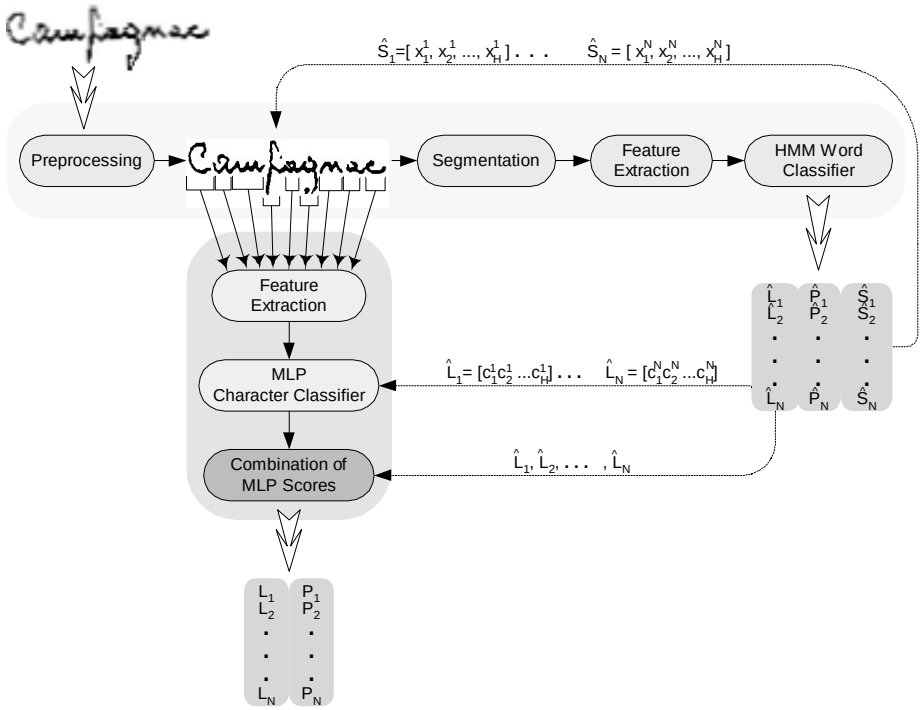

Figure 1. An overview of the HMM/NN hybrid handwriting recognition system.

segment x_h given that the character class c_h was assigned to such a segment. The output of the character classifier is only a probability score (*a posteriori* probability) for the given character class, $P(c_h|x_h)$.

The features extracted from the segmented characters consist of global (top, bottom, left and right profiles and horizontal and vertical projection histograms) and local (contour–directional histogram) features. The former features are extracted from the whole character and normalized to 10 bins at each axis to have an equal number of elements for all characters. The latter is extracted from a 3×2 grid. The combination of the features yields a 108–dimensional feature vector. These features were selected among several others in an empirical evaluation that was previously carried out on the NIST database.

We have designed a simple unconstrained character recognizer based on a multilayer perceptron (MLP). The choice of such a classifier to perform the character recognition task was determined by several constraints such as: recognition speed, capacity of dealing with unbalanced distribution of samples per class, and mainly because, if properly configured, an MLP classifier estimates Bayesian *a posteriori* probabilities [8]. The last characteristic is very important, because later we want to combine the output of the character recognizer with that of the HMM classifier. If both classifiers estimate Bayesian *a posteriori* probabilities at the output, they can be combined in a probabilistic framework [12].

3.1 Problems of Character Omission

The HMM model that we have used [4] includes null transitions that model undersegmentation problems or the absence of a character within a word (usually due to the misspelling of the word by the writer). So, our problem is to estimate an *a posteriori* probability if we do not have a feature vector. We have analyzed different ways to overcome such a problem and the alternative which produced the best results is to use an average *a posteriori* probability score for each character class which is obtained by using the MLP as a standard classifier. For each character class we compute the average when the class is correctly recognized by the neural classifier.

3.2 Combination of the Character Scores

Having the scores of each character segmented from a word hypothesis, we can combine such scores to obtain the score for the word hypothesis. So, the output of the character recognizer denoted as Ψ_{NN}, can be represented by:

$$\Psi_{\text{NN}} = P(L_n|S_n) = \prod_{h=1}^{H} \frac{P(c_h|x_h)}{P(c_h)} P(L_n) \qquad (4)$$

where $P(L_n|S_n)$ is the *a posteriori* probability of the word hypothesis L_n, $P(c_h|x_h)$ is the *a posteriori* probability estimated by the NN to each segment x_h, $P(c_h)$ is the *a priori* probability of the character class h and it can be estimated from the words in the recognition vocabulary, and $P(L_n)$ is the probability of the word hypothesis n. $P(L_n)$ can be neglected since the words in the lexicon have the same *a priori* probability. Figure 2a shows an example of N–best list where the information of the segmentation is used to generate the characters of Figure 2d. The scores assigned by the character classifier and the resulting word scores are also shown.

4 Combination of the HMM and NN Outputs

The methods that can be used to combine multiple classifier decisions depend on the types of information produced by the individual classifiers. The HMM classifier produces at the output a ranked list of words with estimated *a posteriori* probabilities. The output of the character classifier is a non–ranked list with the same word hypotheses, however, with different scores that estimate *a posteriori* probabilities assigned to each hypothesis.

Given that both classifiers estimate *a posteriori* probability at the output, we can obtain a composite score P_{total} by a weighted combination of the outputs of both classifiers:

$$P_{\text{total}} = \alpha \log(\text{HMM}_{\text{score}}) + \beta \log(\text{NN}_{\text{score}}) \qquad (5)$$

where α and β are the weights associated to the HMM and to the NN classifier respectively which sum to one.

So, we end up with a rescored N–best list that can be reordered based on the composite scores. It is expected that the truth hypothesis be shifted up to the top of the list. Since both classifiers already generate estimates of *a posteriori* probabilities, the output probability scores were not normalized before the combination, instead the raw scores provided by both classifiers were used. The attempts to use normalized scores did not bring any improvement to the final results. Another remark is the normalization of the scores with the length of the word hypothesis was neglected since when large vocabularies are used, the word hypotheses in the N–best list tend to have approximately the same number of characters. However, this issue must be considered in the case of N–best lists with high variation in the word lengths.

5 Experiments

Experiments were carried out on two databases: SRTP and NIST. The SRTP database contains more than 20,000 digitized images of postal envelopes from where 20,193 words corresponding to city names were extracted and further split into three datasets: 12,049 words for training, 3,470 for validation and 4,674 for testing. A character database was built automatically from the three datasets. For the training and validation sets, we have selected the words correctly recognized by the HMM classifier in a very selective task where an 85,000–word lexicon was used. Based on the segmentation of such words into characters given by the backtracking of the HMM decoding algorithm, segments representing characters were recovered as well as their classes. No cleaning or visual inspection of the resulting character quality was done. The NIST database was also used in selecting features and to determine the architecture of the NN classifier.

5.1 HMM Classifier

The handwritten word recognition system is composed of several modules: pre–processing, segmentation, feature extraction, training and recognition. The pre–processing normalizes the word images in terms of slant and size. After, the images are segmented into graphemes and the sequence of segments is transformed into a sequence of symbols. There is a set of 70 classes among characters (26 uppercase and 26 lowercase), digits (10) and special symbols (8) that are modeled by 10–state transition–based HMMs with forward and null transitions [4]. The character HMMs were trained using the maximum likelihood criterion and the Baum–Welch algorithm.

The word recognizer performs a huge task of classifying a pattern to one of the possible V classes. The lexicon contains city names that may be single or compound words such as “Chire en Montreuil”. The average length of the words is 12.12 characters, and the shortest and the longest words have 2 and 56 characters respectively. Recognition is carried out by a lexicon–driven search based on a fast two–level HMM decoding algorithm [6].

5.2 NN Classifier

The NN classifier is based on an MLP with 3 layers. To ensure that the output of the classifier estimates Bayesian *a posteriori* probabilities we have used a squared–error cost function and sigmoidal nonlinearities trained with back-propagation. The architecture of the MLP classifier (108–90–26) was obtained experimentally based on the character recognition rates achieved on the training, validation and testing sets of the NIST database (Table 1). Training used 1,660 and 1,440 samples per uppercase (A–Z) and lowercase (a–z) character class respectively. However uppercase and lowercase representations of a character were merged to form a metaclass. Results for the three datasets are shown in Table 1.

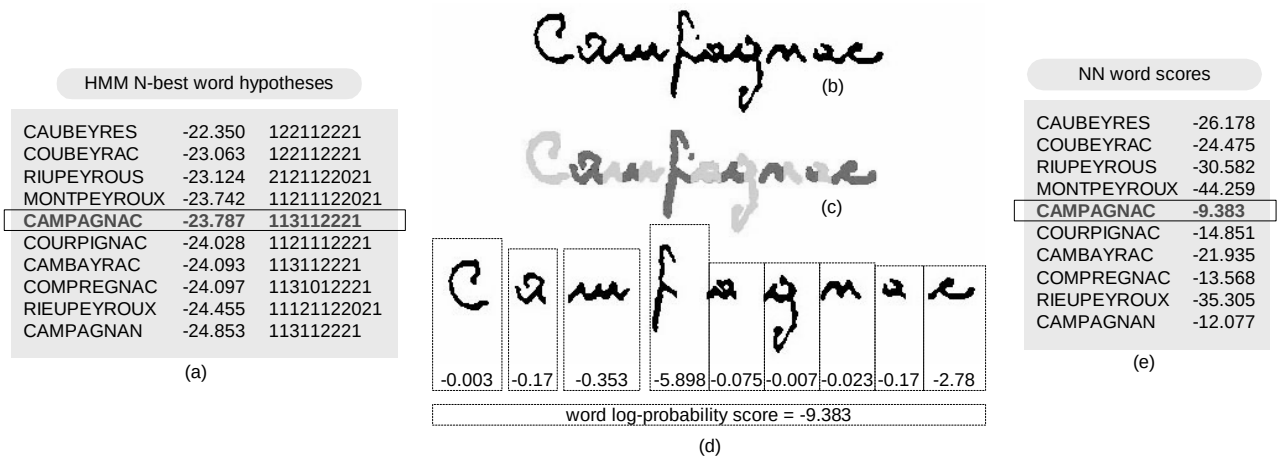

Figure 2. (a) List of N–best word hypotheses generated by the HMM classifier with the log–probability scores and segmentation; (b) Input handwritten word; (c) Loose segmentation of the word into graphemes produced by the segmentation module; (d) Final segmentation of the word into characters obtained from the information provided at the output of the HMM classifier, log–probability scores assigned by the NN classifier to each segment, and the composite score obtained by combining the segment scores, (e) List of N–best word hypotheses scored by the NN classifier.

Further, the same classifier was trained with the characters segmented from the words in the SRTP database. However, for such a database, some classes (such as vowels) had several thousand samples, while others (such as "k" and "w") had less than 10. We balanced the frequency of classes (1,630 samples per class) during the training by skipping and repeating patterns based on a repetition factor [11]. Each repeated character was randomly changed in skew, rotation, and scale. *A priori* class distribution correction was applied to the output of the MLP [8]. Results for the three datasets are shown in Table 1.

5.3 Hybrid HMM/NN Classifier

Experiments have been carried out with unconstrained handwritten words of the SRTP database. Table 2 shows the recognition rates obtained by using the HMM classifier alone, the NN classifier alone, and the combination of HMM/NN classifier on the testing dataset of 4,674 unconstrained words.

6 Conclusions

In this paper we have presented a simple scheme for the integration of two different classifiers. The NN classifier uses the segmentation produced by the HMM classifier to identify characters within the word hypotheses and to assign *a posteriori* probability to each of them. Word scores are obtained by the product of the scores (or sum of the log–scores) of the individual characters that form a word hypothesis. The scores produced by the HMM and the NN classifiers are combined in a probabilistic framework and the N–best word hypothesis list is reordered, shifting up the hypotheses with higher scores. This hybrid classification scheme provides an improvement of 6.7–9.4% in the word recognition rate for the top word hypothesis. Other combination strategies such as majority vote and borda count were also investigated, however the results achieved are below from those obtained by the weighted rule.

The results obtained so far are encouraging, and many points still require attention such as improvements on the feature set used with the NN classifier since it was developed on the NIST dataset as well as the MLP architecture. This may be one of the reasons why the performance to recognize segmented characters is not so good (73.51%). Another point that must be highlighted is that no visual inspection of the data resulting from the segmentation of words into characters was done. We have simply trusted the HMM classifier as performing a good character segmentation. Of course there are segmentation and labelling problems, however even ignoring them, we have achieved very promising results.

Acknowledgments

The authors would like to acknowledge the CNPq–Brazil (grant refs 200276–98/0) and the MEQ–Canada for supporting this research, and the Service Technique de la Poste

Table 1. Size of the datasets and isolated character recognition rates for the NIST and SRTP databases.

Dataset	NIST*		SRTP	
	Number of Samples	Recognition Rate (%)	Number of Samples	Recognition Rate (%)
Training	80,600	94.71	84,760	76.48
Validation	23,670	89.98	27,282	73.54
Test	16,900	88.10	34,731	73.51

*Training: hsf_0, hsf_1, hsf_2, hsf_3; Validation: hsf_7; Testing: hsf_4

Table 2. Results for word recognition using HMM alone, NN alone and the combination of both classifiers for 3 sizes of lexicons: 10,000, 40,000, and 80,000 words.

Lexicon Size	Classifier	Recognition Rate (%)			
		TOP1	TOP2	TOP5	TOP10
10,000	HMM	81.06	85.83	90.58	92.36
	NN	80.82	85.96	90.27	—
	HMM+NN	87.76	90.09	91.98	—
40,000	HMM	73.23	79.48	84.64	87.91
	NN	74.19	80.28	85.12	—
	HMM+NN	81.30	84.85	86.54	—
80,000	HMM	68.65	75.50	81.32	85.10
	NN	71.41	77.73	82.49	—
	HMM+NN	78.05	82.03	84.04	—

(SRTP–France) for providing us the database and the baseline recognition system.

References

[1] F. Camastra and A. Vinciarelli. Cursive character recognition by learning vector quantization. *Pattern Recognition Letters*, 22:625–629, 2001.

[2] M. Y. Chen, A. Kundu, and S. N. Srihari. Variable duration hidden markov model and morphological segmentation for handwritten word recognition. *IEEE Transactions on Image Processing*, 4(12):1675–1688, 1995.

[3] S. J. Cho, J. Kim, and J. H. Kim. Verification of graphemes using neural networks in an hmm–based on–line korean handwriting recognition system. In *Proc. 7th International Workshop on Frontiers in Handwriting Recognition*, pages 219–228, Amsterdam, Netherlands, 2000.

[4] A. El-Yacoubi, M. Gilloux, R. Sabourin, and C. Y. Suen. Unconstrained handwritten word recognition using hidden markov models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):752–760, 1999.

[5] L. Heutte, T. Paquet, J. V. Moreau, Y. Lecourtier, and C. Olivier. A structural/statistical feature based vector for handwritten character recognition. *Pattern Recognition Letters*, 19:629–641, 1998.

[6] A. L. Koerich, R. Sabourin, and C. Y. Suen. Fast two–level viterbi search algorithm for unconstrained handwriting recognition. In *Proc. 27th International Conference on Acoustics, Speech, and Signal Processing*, Orlando, USA, 2002. To appear.

[7] A. L. Koerich, R. Sabourin, C. Y. Suen, and A. El-Yacoubi. A syntax–directed level building algorithm for large vocabulary handwritten word recognition. In *Proc. 4th International Workshop on Document Analysis Systems*, pages 255–266, Rio de Janeiro, Brasil, 2000.

[8] M. D. Richard and R. P. Lippmann. Neural network classifiers estimate bayesian a posteriori probabilities. *Neural Computation*, 3:461–483, 1991.

[9] A. Senior. *Off–Line Cursive Handwriting Recognition using Recurrent Neural Networks*. PhD thesis, University of Cambridge, Cambridge, England, September 1994.

[10] F. Wang, L. Vuurpijl, and L. Schomaker. Support vector machines for the classification of western handwritten capitals. In *Proc. 7th International Workshop on Frontiers in Handwriting Recognition*, pages 167–176, Amsterdam, Netherlands, 2000.

[11] L. Yaeger, R. Lyon, and B. Webb. Effective training of a neural network character classifier for word recognition. In *Proc. Advances in Neural Information Processing Systems*, pages 807–813, Seattle, USA, 1997.

[12] G. Zavaliagkos, Y. Zhao, R. Schwartz, and J. Makhoul. A hybrid segmental neural net/hidden markov model system for continuous speech recognition. *IEEE Transactions on Speech and Audio Processing*, 2(1):151–160, 1994.