An Analytical Handwritten Word Recognition System with Word-level Discriminant Training

Yong Haur Tay[1], Pierre-Michel Lallican[2], Marzuki Khalid[1], Stefan Knerr[2], Christian Viard-Gaudin[3]
[1]Centre for Artificial Intelligence & Robotics (CAIRO),
Universiti Teknologi Malaysia, Jalan Semarak, 54100 Kuala Lumpur, Malaysia.
{yhtay,marzuki}@utmkl.utm.my
[2]Vision Objects, 9, Rue du Pavillon, 44980 Sainte Luce, France.
{pmlallican, stefan.knerr}@visionobjects.com
[3]Laboratoire IRCCyN/UMR CNRS, Ecole Polytechnique de l'Université de Nantes
Rue Christian Pauc, BP 60601, 44306 Nantes Cedex 03, France.
cviard@ireste.fr

Abstract

This paper describes an analytical handwritten word recognition system combining Neural Networks (NN) and Hidden Markov Models (HMM). Using a fast left-right slicing method, we generate a segmentation graph that describes all possible ways to segment a word into characters. The NN computes the observation probabilities for each character hypothesis in the segmentation graph. Then, using concatenated character-HMMs, a likelihood is computed for each word in the lexicon by multiplying the observation probabilities over the best path through the graph. The role of the NN is to recognize characters and to reject non-characters. We present our approach to globally train the word recognizer using isolated word images. Using a Maximum Mutual Information (MMI) cost function at the word level, the discriminant training updates the parameters of the NN within a global optimization process based on gradient descent. The recognizer is bootstrapped from a baseline recognition system, which is based on character level training. The recognition performances of the globally trained system are compared to the baseline system.

1. Introduction

Most of today's state-of-the-art cursive word recognizers combine the discriminative power of NNs with the excellent capacity of modeling sequences provided by HMMs [1, 2, 3, 4, 5]. The role of the NN in these hybrid systems is to provide probabilities for character hypothesis or sub-character entities, whereas the HMM is used to model the sequence of observations and to compute word likelihoods, usually based on a lexicon. This approach often yields better recognition performances compared to conventional discrete or continuous HMM recognizers [6]. However, there are some limitations of these systems:

1. The NN and HMM are usually trained separately. In earlier publications [3, 6], we have presented an approach in which the NN has been trained by a gradient descent algorithm using isolated characters that have been segmented from the training word database; either by hand, or by the HMM using Viterbi alignment. The transition probabilities of the HMMs have been estimated using the Baum-Welch algorithm within a Maximum Likelihood training scheme, but the objective function used for training the NN was not directly linked to the word recognition performance.
2. Within this type of approach, the outputs of the NN are usually divided by the class prior probabilities. This results in 'scaled-likelihood', which are used as the observation probabilities in the HMMs. This normalization often generates problems if some character classes have very low prior probabilities. For instance capital letters usually have relatively small priors (thus large scaled-likelihoods) compared to lower-case letters.
3. '*Junk*' problem. Within the over-segmentation approach, the NN needs to handle character hypothesis that are not characters, i.e. they only represent part of a character or a combination of characters. Therefore, we need to provide non-character samples ('*junk*' samples) to train the NN. It is not obvious how to generate these non-character training examples, nor which and how many of them should be used for training.

To avoid these limitations, it is possible to globally train the recognizer using a word-level objective function and a gradient descent algorithm [4, 5]. In this paper, we show (i) how we can directly use the NN outputs as the observation probabilities, without dividing them by the character class priors, and (ii) how to train the NN in order to reject non-characters implicitly, without explicitly creating junk samples.

This paper is organized as follows. We first give an overview of our cursive handwritten word recognizer in Section 2. In section 3, we present the proposed method to globally train the NN using a word-level objective function. In section 4, the recognition performance of the globally trained recognizer is compared to a recognizer trained with a character-level objective function. These results have been obtained using the IRONOFF [7] and AWS [8] isolated word database.

2. NN-HMM Hybrid Recognition System

Our cursive handwritten word recognizer is lexicon driven, and it is based on an over-segmentation of the word image into vertical slices. A detailed description of the system can be found in [3]. The recognition process can be divided into 4 steps: preprocessing, over-segmentation, character recognition using NNs, and word likelihood computation using HMMs.

2.1. Pre-processing

First, the binarized word image is slant corrected. The average slant of a word is estimated using the chain code histogram of all contour pixels. Horizontal shearing corrects the slant of the image. This is followed by reference line detection, which define the position of the core-zone in the word. To detect the reference lines, we first smooth the contours of the binary image. Then, we extract the local minima and maxima of the internal and external contours of the word image. Together with a priori probability distributions on the vertical line positions, these extrema are used by an Expectation-Maximization (EM) algorithm in order to estimate the position and the slant of 4 straight parallel reference lines [9]. The reference lines are then used to normalize skew and size of the handwriting, as well as to retrieve ascenders and descenders of characters.

2.2. Segmentation

The word image is cut from left to right into slices of variable width. In order to achieve size invariant segmentation, the average slice width depends on the height of the core-zone of the word. The parameters of the segmentation algorithm are chosen such that each slice contains either a character or part of a character, but not more than a character (over-segmentation). The exact position of each cut is determined such that a minimum number of ink pixels have to be crossed. From the left-right ordered sequence of slices, we combine several consecutive slices to form character hypothesis. A segmentation graph represents all possible ways to segment the word into characters. Fig. 1 depicts an example of a word image cut into four slices. The right side of the figure shows the segmentation graph that consists of character hypothesis composed of 1 to 4 consecutive frames. A maximum of 9 consecutive frames is allowed for character hypothesis.

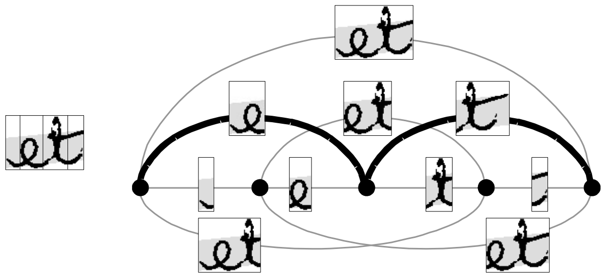

Fig. 1. (Left) Word 'et' is segmented in 4 slices. (Right) Segmentation graph describing all possible ways to segment the word into characters. The bold lines indicate the true segmentation paths for the word 'et'.

2.3. Character Recognition using NN

For each character hypothesis in the segmentation graph, we perform character recognition. We first extract geometrical features from the character hypothesis and feed them into a trained NN for classification. The topology of the NN is a 2-layered Perceptron (MLP), with Softmax activation functions at the output layer, which provide probabilities for the character classes. Notice from Fig. 1 that since there are non-character hypothesis ('*junk*'), the NN must be able to model these. It has to provide low probabilities for all other character classes except the '*junk*' class. Therefore we add a '*junk*' output neuron to the NN.

Note that it would be possible to use sigmoid output functions in the NN instead of the Softmax functions and to train the NN to provide zero probabilities for all outputs when 'junk' is presented. However, we have found experimentally that this approach needs larger training sets than the first one in order to obtain comparable results. It seems therefore preferable to explicitly model the '*junk*' class.

2.4. Word-likelihood Computation using HMM

For each entry in the lexicon, we build a word-HMM by concatenating character-HMMs and ligature-HMMs as illustrated in Fig. 2. The gray states are 'dummy' states that do not emit observation probabilities. They allow for the alignment with the observation sequence where each slice is an observation, and thereby allow the use of the usual HMM tools [10]. The observation probabilities in each emitting state of the basic-HMMs (character- and ligature-HMMs) are computed by the NN. All transition probabilities are set to 1 and are not modified during training. The likelihood for each word in the lexicon is computed by multiplying the observation probabilities over the best path through the graph using the Viterbi algorithm. The word-HMM with the highest probability is the first recognition candidate.

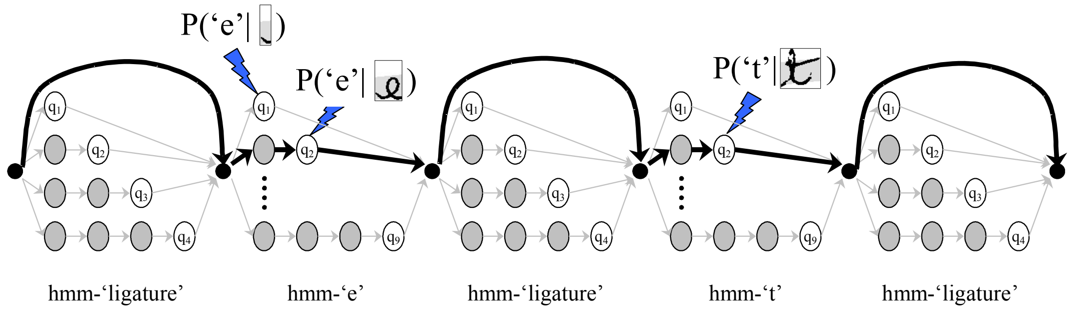

Fig. 2. Word-HMM 'et' is composed of character-HMMs 'e' and 't', with ligature-HMMs inserted before and after. Lines in bold indicate the best path through the model for the example of Fig. 1.

3. Word-Level Discriminant Training

Most of the NN-HMM hybrid systems used for cursive handwriting recognition train the NN and HMM separately, i.e. by segmenting words into isolated characters, which are then used to train the NN [1, 6]. Therefore, the NN is optimized at the character level, which does not guarantee optimal recognition performance at the word level. Furthermore, character level training implies that we need to provide examples of characters as well as non-characters ('*junk*') to train the NN, which is not an easy task [3].

Word level discriminant training seems to be an answer to our problem [2, 4, 5]. Instead of generating isolated characters from the word images in order to train the NN separately, we instantly back-propagate the error at the word level into the NN to update its parameters. All transition probabilities in the HMMs are set to 1 and are not modified during training (see Fig. 2).

We base the word level objective function L on the Maximum Mutual Information (MMI):

$$L = \log \frac{P(O \mid \lambda^{\tau})}{\sum_{\lambda'} P(O \mid \lambda')} = \log P(O \mid \lambda^{\tau}) - \log \sum_{\lambda'} P(O \mid \lambda') \quad (1)$$

where $P(O \mid \lambda)$ is the likelihood of the observation sequence $O = O_1 O_2 \ldots O_T$, given the HMM λ. λ^{τ} is the true word-HMM, and λ' varies over all word-HMMs in the given lexicon. The objective function is optimized using gradient descent (back-propagation).

To reduce the computational complexity of Eq. (1), an approximate cost of L can be written as

$$L' = \log P(O \mid \lambda^{\tau}) - \log P(O \mid \lambda^{*}) \quad (2)$$

where λ^{*} is the HMM with the largest likelihood among all λ' or

$$\lambda^{*} = \arg\max_{\lambda'} P(O \mid \lambda') \quad (3)$$

Eq. (2) states that the word-level objective function is based on the difference between the log-likelihood of the true HMM, λ^{τ} and the best HMM, λ^{*}. If λ^{τ} is also the best HMM, then $L' = \log P(O \mid \lambda^{\tau}) - \log P(O \mid \lambda^{\tau}) = 0$, and thus, no error is back propagated to update the NN. Otherwise, we change the weights, W of the NN using the chain rule:

$$\frac{\partial L'}{\partial W} = \sum_{j} \frac{\partial L'}{\partial b_j(O_t)} \cdot \frac{\partial b_j(O_t)}{\partial W} \qquad \forall O_t \quad (4)$$

The word likelihood of HMM λ, given observation sequence O can be defined as:

$$P(O \mid \lambda) = \sum_{\Gamma} \prod_{t=1}^{T} a_{q_{t-1}q_t} b_{q_t}(O_t) \quad (5)$$

where a, $b(O_t)$ are the transition probabilities, and observation probabilities, respectively, and the sum runs over all paths Γ through the HMM λ [11]. In this work, transition probabilities, $a_{q_{t-1}q_t} = 1$. The derivatives of $b(O_t)$ can be written as:

$$\frac{\partial b_{q_t}(O_t)}{\partial b_j(O_t)} = \delta_{j,q_t} = \delta_{j,q_t} \cdot \frac{b_{q_t}(O_t)}{b_j(O_t)} \quad (6)$$

where $\delta_{i,j} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$

Therefore, the derivative of the word likelihood with respect to the observation probability can be written as:

$$\begin{aligned}\frac{\partial P(O \mid \lambda)}{\partial b_j(O_t)} &= \sum_{\Gamma}\prod_{t=1}^{T} a_{q_{t-1}q_t}\delta_{j,q_t}\cdot\frac{b_{q_t}(O_t)}{b_j(O_t)} \\ &= \frac{1}{b_j(O_t)}\sum_{\Gamma}\prod_{t=1}^{T} a_{q_{t-1}q_t}\delta_{j,q_t} b_{q_t}(O_t) \\ &= \frac{1}{b_j(O_t)}\sum_{\Gamma} P(O,\Gamma,q_t = j \mid \lambda) \\ &= \frac{1}{b_j(O_t)} P(O,q_t = j \mid \lambda)\end{aligned} \tag{7}$$

where $P(O,q_t = j \mid \lambda)$ can be computed by a dynamic programming algorithm [11].

The first term in Eq. (4) can be further derived as:

$$\begin{aligned}\frac{\partial L^{'}}{\partial b_j(O_t)} &= \frac{1}{P(O \mid \lambda^{\tau})}\cdot\frac{\partial P(O \mid \lambda^{\tau})}{\partial b_j(O_t)} - \frac{1}{P(O \mid \lambda^{*})}\cdot\frac{\partial P(O \mid \lambda^{*})}{\partial b_j(O_t)} \\ &= \frac{1}{P(O \mid \lambda^{\tau})}\cdot\frac{P(O,q_t = j \mid \lambda^{\tau})}{b_j(O_t)} - \frac{1}{P(O \mid \lambda^{*})}\cdot\frac{P(O,q_t}{b_j(} \\ &= \frac{1}{b_j(O_t)}\left(\frac{P(O,q_t = j \mid \lambda^{\tau})}{P(O \mid \lambda^{\tau})} - \frac{P(O,q_t = j \mid \lambda^{*})}{P(O \mid \lambda^{*})}\right)\end{aligned} \tag{8}$$

where $\sum_j P(O,q_t = j \mid \lambda) / P(O \mid \lambda)$ is the probability that a given observation is one of the characters in the considered word-HMM λ.

For the second term of the Eq (4), as $b_j(O_t)$ is actually the output of the NN given O_t, we can use the usual back-propagation algorithm to update the weights of the NN.

Fig. 3 illustrates the basic idea of the global training algorithm. The error signal back-propagated into the NN is based on the global word level objective function, and therefore, modifies the outputs corresponding to the character classes involved in the true HMM and the best HMM. The '*junk*' class is only trained indirectly through the Softmax constraint (sum over all outputs equals 1), and thereby absorbs the probability mass corresponding to patterns that are not well explained by the character classes. Unlike with our previous NN-HMM hybrid, it is not necessary to divide the NN outputs by the character class priors in order to obtain probabilities that are based on approximately equal priors. In the present approach, the importance of each character class is determined automatically by the word level training, and can be different for each character class depending on its discriminant power at the word level.

4. Experiments and Results

We have tested our recognition system on isolated words from the IRONOFF database, which contains a total of 36,396 isolated French word images from a 196 word lexicon [7]. The training data set contains 24,177 word images, and another 12,219 images have been used for tests. All writers in the training data set are different from the writers in the test data set. Therefore, the recognition rates reflect an omni-scriptor situation. To further challenge the recognizer, we have also performed tests using a 2000 word lexicon for which we have chosen words in order to maximize the risk of confusion.

Another database AWS, collected by Senior and Robinson [8] consists of a 1374 word lexicon with 2360, 675 and 1016, training, validation and test images, respectively. These 4035 words have been extracted from a text that belongs to the LOB Corpus and was written by single scriptor.

Recognition scores at the word level have been evaluated using two performance measures: the top 1 recognition rate Rec(1), and the average position $\overline{pos}$ of the true class in the candidate list.

The experiments have been carried out as follows. First, we have trained a baseline recognizer using character level training. Then, using the segmented characters generated by the baseline recognizer, we have trained the NN at the character level. Once the NN has been bootstrapped, we started to train the recognizer using the word level discriminant training.

Results in Table 1 show the superior performance of the word-level training on all the experiments as compared to the local training. Our result on the AWS data can be compared to the result by Senior [8] who has obtained a 93.4% recognition rate, and to 83.6% presented by Vinciarelli [12].

5. Conclusions

In this paper, we have described an offline handwriting recognition system that is globally trained using a word level objective function. Recognition results

Table 1. Recognition performances of the recognizers on three test data sets.

Database	Lexicon Size	Char-level Training		Word-level Training	
		Rec(1)	$\overline{pos}$	Rec(1)	$\overline{pos}$
IRONOFF	196	96.1	1.4	97.1	1.2
IRONOFF	2000	83.1	5.9	88.1	3.0
AWS	1374	93.6	3.0	94.6	1.3

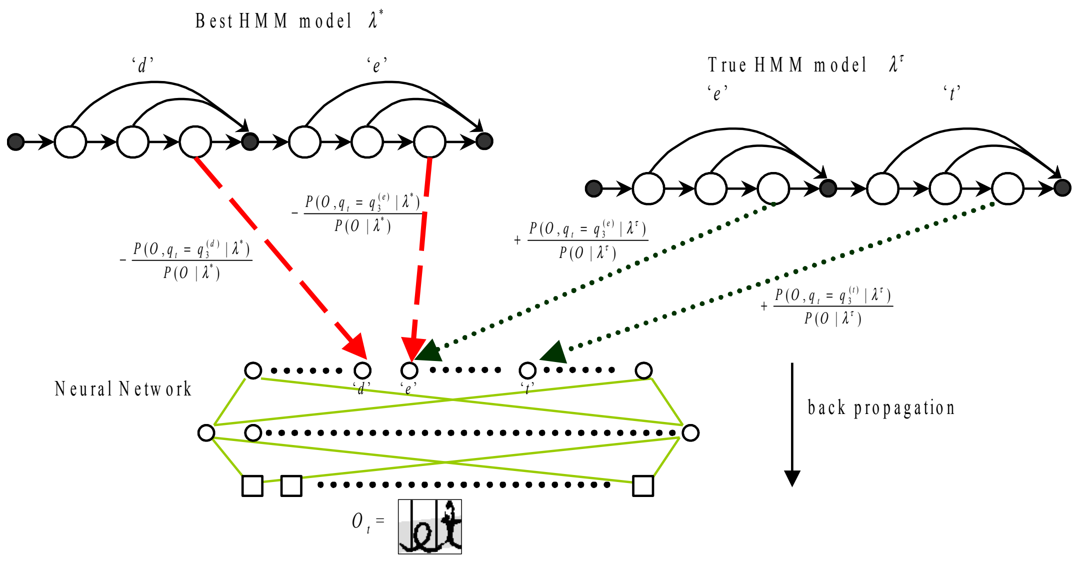
Best HMM model λ^*
'd'
'e'
True HMM model λ^τ
'e'
't'
$-\frac{P(O,q_t=q_3^{(d)}\mid\lambda^*)}{P(O\mid\lambda^*)}$
$-\frac{P(O,q_t=q_3^{(e)}\mid\lambda^*)}{P(O\mid\lambda^*)}$
$+\frac{P(O,q_t=q_3^{(e)}\mid\lambda^\tau)}{P(O\mid\lambda^\tau)}$
$+\frac{P(O,q_t=q_3^{(t)}\mid\lambda^\tau)}{P(O\mid\lambda^\tau)}$
Neural Network
'd'
'e'
't'
back propagation
O_t =

Fig. 3. For each observation O_t, positive and negative gradients from the true HMM and the best HMM, respectively, are back-propagated into the NN.

on isolated words from the IRONOFF and AWS database have been presented, which show the superiority of this approach compared to character level training. Further experiments will be carried out to train the recognizer directly from a random initialization, thereby eliminating the need to bootstrap the recognizer.

Acknowledgements

This research has been partly funded by the Ministry of Science, Technology and Environment (MOSTE), Malaysia, through IRPA Grant 72903 and the French government through the French Embassy in Kuala Lumpur.

References

[1] R.Plamondon, S.N.Srihari, "On-Line and Off-line Handwriting Recognition: A Comprehensive Survey", IEEE Transactions on PAMI, Vol.22, No. 1, pp.63-84, 2000.

[2] Y.LeCun, L.Bottou, Y.Bengio, P.Haffner, "Gradient-Based Learning Applied to Document Recognition", Proceedings of IEEE, Vol. 86, No. 11, pp. 2278-2324, 1998.

[3] Y.H.Tay, P.M.Lallican, M.Khalid, C.Viard-Gaudin and S.Knerr, "An Offline Cursive Handwritten Word Recognition System", IEEE Region 10 International Conference on Electrical and Electronic Technology (TENCON), 2001.

[4] A.Krogh, S.K.Riis, "Hidden Neural Networks", Neural Computation, 11, pp. 541- 563, 1999.

[5] Y. Bengio, R. De Mori, G. Flammia, R. Kompe, "Global Optimization of a Neural Network-Hidden Markov Model Hybrid", IEEE transactions on Neural Networks, Vol. 3, No. 2, pp. 252-258, 1992.

[6] S.Knerr, E.Augustin, "A Neural Network-Hidden Markov Model Hybrid for Cursive Word Recognition", International Conference on Pattern Recognition 98, Brisbane, 1998.

[7] C.Viard-Gaudin, P.M.Lallican, S.Knerr, P.Binter, "The IRESTE On/Off (IRONOFF) Dual Handwriting Database", International Conference on Document Analysis and Recognition, 1999.

[8] A.W.Senior and A.J.Robinson, "An Off-Line Cursive Handwriting Recognition System", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, 1998.

[9] Y.Bengio, Y.LeCun, "Word Level Training of a Handwritten Word Recognizer Based on Convolutional Neural Networks", International Conference on Pattern Recognition, pp. 88-92, 1994.

[10] M.Y.Chen, A.Kundu, "An Alternative to Variable Duration HMM in Handwritten Word Recognition", International Workshop on Frontiers in Handwriting Recognition III, Buffalo, 1993.

[11] L.R.Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", Proceedings of the IEEE, vol. 77, pp. 257-285, 1989.

[12] A. Vinciarelli, J. Luettin, "Off-Line Cursive Script Recognition Based on Continuous Density HMM", International Workshop on Frontiers in Handwriting Recognition, IWFHR'2000, Amsterdam, The Netherlands, Sept 2000, pp. 493-498.