A Deep HMM model for multiple keywords spotting in handwritten documents

Simon Thomas, Clement Chatelain, Laurent Heutte, Thierry Paquet, Yousri Kessentini

Noname manuscript No.
(will be inserted by the editor)

A Deep HMM model for multiple keywords spotting in handwritten documents

Simon Thomas, Clément Chatelain, Laurent Heutte, Thierry Paquet, Yousri Kessentini

the date of receipt and acceptance should be inserted later

Abstract In this paper, we propose a query by string word spotting system able to extract arbitrary keywords in handwritten documents, taking both segmentation and recognition decisions at the line level. The system relies on the combination of a HMM line model made of keyword and non-keyword (filler) models, with a deep neural network (DNN) that estimates the state-dependent observation probabilities. Experiments are carried out on RIMES database, an unconstrained handwritten document database that is used for benchmarking different handwriting recognition tasks. The obtained results show the superiority of the proposed framework over the classical GMM-HMM and standard HMM hybrid architectures.

Keywords Handwriting Recognition · Keyword Spotting · Hidden Markov Models · Deep neural network · Hybrid Deep architecture · DNN HMM

1 Introduction

Recently, several systems for spotting words in unconstrained handwritten documents have been developed [1–7]. Such word spotting systems aim at detecting and recognizing the queried words in document images. They can be applied simply for querying textual handwritten documents, but such systems may also be a component for indexing handwritten document databases by spotting the multiple keywords on which indexation is performed, following the popular bag of word indexation scheme [8].

Formally, one can distinguish *word spotting* approaches and *keyword spotting* approaches. The former aims at detecting a single word in a collection of document images in order to retrieve the subset of document images that contain this specific word. On the contrary, the latter aims at detecting, in a single document, a set of informative words that allow to predict its content or its category using the popular bag of words paradigm for example. These later approaches are used for automatic document categorisation [8,9] for which a reduced lexicon of keywords is needed so as to discriminate between a set of predefined categories of documents. In the literature, most of the word spotting approaches follow the former approach [1–7], but very few of them are designed for multiple keywords spotting [8,10]. Even if the two tasks (word spotting vs. keyword spotting) are used in a different context, they rely on the same handwriting detection/recognition methods and both can be used to perform a quick, writer-independent detection and identification of words in document images.

Word spotting approaches proposed in the literature fall into the two following categories. On one hand, image based methods, also known as "query-by-example", operate through the image representation of the words [1,11–13]. On the other hand, recognition based methods, or "query-by-string" methods, operate with an ASCII representation of the keywords [2,4,8,6]. The first approach has the major drawback to be writer-dependent since it is based on a matching distance between the query image and the word images in the document. It is therefore limited to single writer applications and cannot be used for content-based document categorization. On the contrary, recognition based approaches are not limited to a single writer, but this is obtained at

C. Chatelain
INSA-Rouen
LITIS EA 4108
Tel.: +33-23-2955210
Fax.: +33-23-2955022
E-mail: clement.chatelain@insa-rouen.fr

the expense of a more complex matching process, derived from conventional handwriting recognition methods. They are thus suitable for either document image retrieval or document image categorization.

Recognition-based word spotting approaches have to deal with four main issues: text line detection, text line segmentation into words, word recognition, and rejection of word hypotheses that do not match the query. A first strategy is based on the sequential application of these four steps, as in [4]. In this work, text lines are segmented into words using heuristics and a distance criterion, then words are recognized, and finally word hypotheses are spotted using normalized scores. The major drawback of this strategy is that the word segmentation errors are often irreversible, and will therefore affect significantly the recognition performance. Moreover, the approach presented in [4] is holistic, so the system is not able to spot words that are not present in the training set. Also, the decision rule to accept or reject a word hypothesis, which is the core of the approach, is difficult to design as it is data-dependent.

A second strategy performs every steps in one single global dynamic programing based optimization process that carry out segmentation/recognition/rejection decisions at the line level, using a line model able to describe both the keywords and the non-keywords [7,10,14]. In [7], a line-based HMM word spotting system is applied to handwritten documents [1]. The approach is based on a line model made of a left and right "filler" model surrounding the keyword model. Line decoding based on Viterbi dynamic programing is performed twice: with and without the keyword, leading to two recognition scores. The likelihood ratio of the two scores allows to decide whether the line contains the keyword or not. The main drawback of this approach is its limited local discriminative abilities as the Gaussian mixture model used to compute emission probabilities is trained in a non-discriminative way.

However, it has been shown that embedding a discriminative stage in HMM model can significantly improve the recognition performance. Discrimination between models can be introduced using either discriminative training [15–17], or by replacing the GMM classifier by a discriminative classifier such as a SVM [18, 19] or a neural network [20–22]. In this study, we propose to combine the modeling ability of HMMs with the discriminative ability of a deep neural network architectures in order to design an efficient "query by string" spotting system that will operate at the line level. This is an extension of our preliminary work [Hidden ref] , where the fully generative GMM-HMM architecture is now replaced by the hybrid deep HMM architecture that significantly improves the performance. We also provide a comparison between pure HMM, standard hybrid MLP HMM and deep HMM approaches, using different feature sets and larger lexicon sizes. We finally show that the deep HMM architecture provides the best results, while avoiding the tedious design of an efficient handcrafted feature set dedicated to handwritten word recognition.

When compared with previous word spotting systems, our approach exhibits several advantages. Firstly, the system is ***lexicon-free***, i.e., it is not restricted to spot some specific keywords for which the system is tuned during a preliminary training stage, but it can rather spot any arbitrary keywords. Secondly, the system is ***segmentation-free***, i.e., text lines are not required to be segmented into words. Thirdly, it embeds a ***discriminative*** classifier which outperforms the standard Gaussian mixture model. Fourthly, the system is able to learn some discriminative features from the raw data. To the best of our knowledge, it is the first time such hybrid HMM / DNN architecture is applied on a handwriting recognition task. Such a deep HMM architecture benefits from both modeling and discriminative skills of these two models. We show on the French public incoming mail RIMES database [23] that this deep HMM architecture significantly improves the keyword spotting results when compared with the pure HMM model. In addition, the system is able to spot multiple keywords at a time through a competitive modeling of keywords by introducing a lexicon of keyword in the model. This is a way to cope with the limitation of several keyword-spotting approaches, especially the one presented in [7], which is only able to spot one keyword at a time. To spot multiple keywords, [7] proposes to iterate the spotting process for each keyword of the keyword lexicon, but the authors do not discuss how to solve conflicts when two detected keywords overlap.

This paper is organized as follows. We introduce our generic statistical text line model in Section 2. In Section 3, we describe the deep HMM architecture. An overview of the whole system is described in Section 4. In Section 5, we report the evaluation of our system on a multiple keywords spotting task. Conclusion and future works are given in Section 6.

2 A generic text line discriminative model for multiple keywords spotting

A keyword spotting system should be able to detect the position of each occurrence of a keyword in a document

[1] An extension of this approach has been proposed in [6], where the HMM is replaced by a combination of a LSTM neural network with a CTC algorithm.

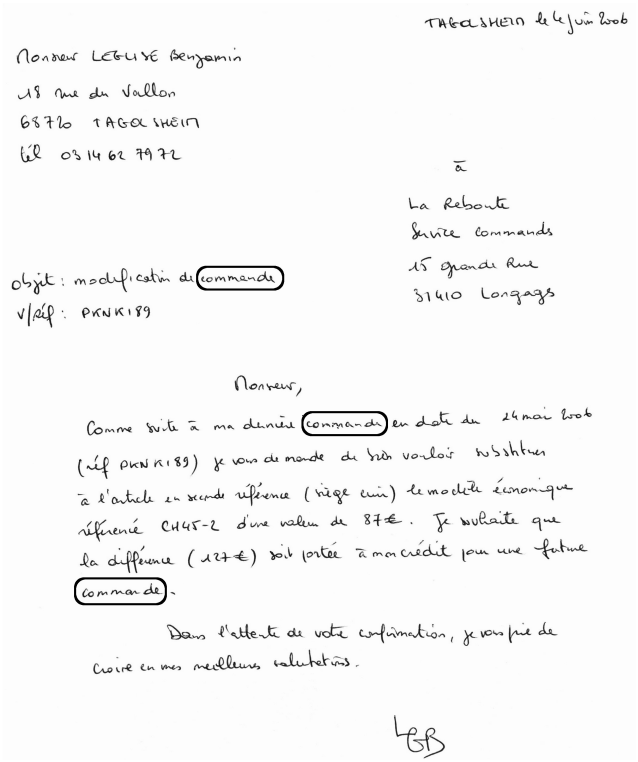


Fig. 1 Expected result for the spotting of the word "commande" in a french handwritten mail.

image, as depicted on Figure 1. Keyword spotting systems rely on the analysis of text lines, in order to decide whether it contains the searched keyword or not. As already mentioned in the introduction, we believe that the segmentation of the text lines into words cannot be reliable enough without recognition, and that a global segmentation/recognition decision should be taken at the line level. A successful strategy in the literature consists in building a text line model in order to allow recognition driven segmentation. For example, in [7,14], a text line model composed of the keyword model surrounded by a filler model is proposed. Line decoding is performed twice, with the text line model and with the filler model, leading to two likelihood scores. Their ratio is used as a decision score compared to a threshold, in order to decide whether a keyword is present or not. In [24], the authors also use a text line model made of the keyword, a "left-extraneous model" and a "right-extraneous model" to spot street names in lines of postal addresses. Once again, the extraneous models act as filler models describing any sequence of characters. The decision is also taken by comparing the global line score to a threshold after normalization.

As we want to spot multiple keywords in one shot, the model should integrate the keyword lexicon, as well as a filler model able to model the irrelevant information (namely Out of Keyword Lexicon (OKL) word sequences and noise). In order to depict a line of text as a sequence of keywords and irrelevant information, a space model must also be integrated in the model.

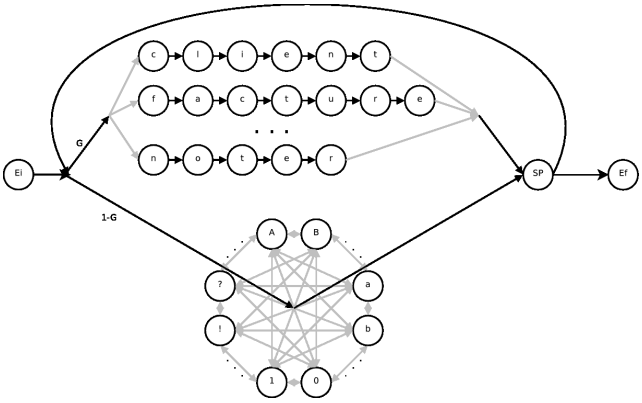


Fig. 2 Global line model containing the keywords lexicon KL (upper part), and the filler model used for modeling everything else (Out of Keyword Lexicon OKL, on the lower part). The line model is also made of a space model ("SP"), and structural initial and final models ("Ei" and "Ef").

Figure 2 represents our model for modeling the two kinds of information that can be encountered within a text line. The Keyword Lexicon (KL) is represented in the upper part of the scheme. It is a parallel model of keywords represented as character sequences that are equally likely to occur in the text. The irrelevant information is represented in the lower part of the Figure. It is used to model any word that does not belong to the Keyword Lexicon, namely the Out of Keyword Lexicon (OKL). It is an ergodic model that accounts for any sequence of alphanumeric characters, including uppercase and lowercase letters, digits and punctuation. Keywords in KL may occur with a lower frequency than words in OKL. This is why the two models compete in parallel in the line model, with parameter G standing for the proportion of relevant information that can appear in the text. Finally, the line model concatenates any KL or OKL words with a white space model before going to the next possible word in the line through the loopy backward transition. The model loops until the end of line is reached. Let us emphasize that as the model simulates a competition between the OKL and each word of the KL, it prevents from the conflicts that may occur between two keywords, or between a keyword and the OKL. It also avoids the use of numerous thresholds for accepting/rejecting the keywords hypotheses. The setting of parameter G is discussed in Section 5.2

As one can see, the handwritten modeling is text line oriented, and fits particularly well with the gen-

erative modeling of Hidden Markov Models (HMM). Indeed, HMM are a well known and established probabilistic tool in sequence modeling [25]. Thanks to an efficient embedded learning through Baum-Welch algorithm, HMM have been widely used for the recognition of handwritten words [26] or sentences [9]. However, despite its appealing generative modeling ability, HMM suffers from a poor discrimination ability at the local frame level. Indeed, the local likelihood $P(o|q_i)$ that stands for the emission probability of observation o when the model is in state q_i is modelled using Gaussian Mixture Models (GMM) that are learned in a generative way. As a consequence, GMM of several states may overlap and are not trained to discriminate between these states. Furthermore, estimating a GMM in high dimension requires a huge amount of training data and one generally limits the dimension of the feature space to prevent from this difficulty. To overcome these limitations, many works have successfully proposed to replace the GMM by a discriminative classifier, most of the time a neural network [20–22,27,28], but SVM have also been proposed [29]. In this paper, we propose to replace the generative stage of HMM by a discriminative stage based on a deep neural network (DNN). This kind of architecture has recently been applied for speech recognition in [30,31] and has shown good performance.

As depicted on Figure 2, two kinds of information must be learnt: the character models (black circles) and the transition probabilities between them (black and grey arrows). Transition probabilities between characters of the KL model are all equal to one as they stand for a lexicon driven recognition. Transition probabilities of the OKL model must be set to build a shallow language model that stands for a lexicon free recognition. Therefore, they are learnt from a general large lexicon. Character models are hybrid left-right DNN-HMM models, where the DNN provides a posteriori probabilities $P(q_i|o)$ that the frame belongs to the HMM state q_i. They require a discriminative training stage on locally labeled data (at the frame level) for the DNN part, while state transition probabilities can be learnt as it is the case for standard HMM using the Baum-Welch algorithm. The main difficulty in training a DNN architecture is to provide a labeled training data at the frame level. This is discussed further in the next section.

3 The deep HMM architecture

Let us consider an observation sequence $O = \{o_1 \dots o_T\}$ of length T, extracted from a line of text L, whatever the nature of the information encoded (a continuous feature vector sequence or simply a sequence of sub images with white and black pixels as will be described further). Let us denote L_{opt} the optimal sequence of labels (KL words and OKL words), that can be associated to O. It can be computed thanks to the equation 1:

$$L_{opt} = \arg\max_L \{P(L|O)\} \tag{1}$$

$$= \arg\max_L \left\{ \frac{P(O|L)P(L)}{P(O)} \right\} \tag{2}$$

$$= \arg\max_L \{P(O|L)P(L)\} \tag{3}$$

where $P(O|L)$ is the probability of a sequence of observations O given the sequence of words L. Even if $P(O)$ is difficult to compute, it is constant and thus does not affect the search of the best state sequence and can be removed from the computation. In equation 3, $P(L)$ is the prior probability of a word sequence. We assume equal priors for every word sequence, and therefore $P(L)$ can be ignored.

Let us assume that L contains N character sequences K_n belonging to KL, M character sequences W_m belonging to OKL, and P spaces S. Then L_{opt} can be found by maximizing over all segmentation/recognition path hypotheses:

$$L_{opt} = \arg\max \left\{ G^N \times \prod_{n=1}^{N} P(O_n|K_n) \times (1-G)^M \times \prod_{m=1}^{M} P(O_m|W_m) \times \prod_{p=1}^{P} P(O_p|S) \right\} \tag{4}$$

where each $P(O|K)$ (respectively $P(O_n|K_n)$, $P(O_m|W_m)$ and $P(O_p|S_p)$) stands for the likelihood of observing the frame sequence O (resp. O_n, O_m and O_p) knowing the character sequence K (resp. K_n, W_m and S). In an HMM framework, these likelihoods are computed at the frame level using the Markov assumption:

$$P(O|K) = \prod_{t=1}^{T} P(o_t|q_i)P(q_i|q_{i-1}) \tag{5}$$

where $\{q_i\}$ are the states involved in the character sequence K, and T is the number of frames. Finally, finding the best character sequence L_{opt} is performed using the Time Synchronous Beam Search algorithm presented in [32], by maximizing over the whole line the quantity $P(O|L)P(L)$. It depends on:

- the model and the *a priori* information, *i.e.* the G parameter and the lexicon constraints modelled by the transition probabilities between states $P(q_i|q_{i-1})$;

– the local discrimination ability of the system $P(o_t|q_i)$.

In order to find the best character sequence, one needs to estimate the local likelihood quantities $P(o_t|q_i)$. A statistical classifier such as a neural network for example generally provides a posteriori probabilities $P(q_i|o_y)$ and the Bayes law is applied to rescale its outputs into local likelihood: $P(o_t|q_i) = \frac{P(q_i|o_t) \times P(o_t)}{P(q_i)}$. During recognition, the term $P(o_t)$ is a constant for all classes and does not change the classification, it can be omited for every q_i. Finally, the scaled likelihood $\frac{P(q_i|o_t)}{P(q_i)}$ can be used as an emission probability for the HMM.

3.1 Discriminative deep architecture

Let us recall that discriminative deep architectures [33] have proven to achieve very high performance on various applications such as digit recognition [34,35], audio classification [36], or action recognition [37]. They are based on the (old) idea that training a neural network with many hidden layers is a good way to *(i)* handle high dimensional input, and *(ii)* estimate complex decision boundaries between classes. The novelty lies in the way the hidden layers are trained. Indeed, when using the standard back-propagation algorithm on many layers, the energy of the errors is too weak to train the whole network [38]. This issue is also known as "the vanishing gradient". Therefore, for training a deep neural network with Λ-hidden layer, a two-step training has been proposed [33,38,39]:

- The first $\Lambda-1$ hidden layers are sequentially trained in an unsupervised way. Each layer is fed by the output of the previous layer and is trained as an auto-associator [2]. These layers are called *model layers*, while their training is called *pre-training*. Taking raw data as input, a high level representation of the data can be obtained at the output of the top layer. This pre-training process can be considered to be an automatic feature extraction process [38].
- Then the last layer (called *decision layer*) is added to the network. It is trained by unlocking all the parameters of the model layers and performing supervised gradient back-propagation on the whole network. This operation allows to learn the discriminant decision function, and is known as *fine-tuning*.

3.1.1 Definition of the network

More formally, let us define a deep network containing Λ layers, where each layer computes its outputs $\mathbf{H}^\lambda$,

[2] Note that a Restricted Boltzman Machine (RBM) can also be used, in this case they are called *Deep Belief Networks* [33].

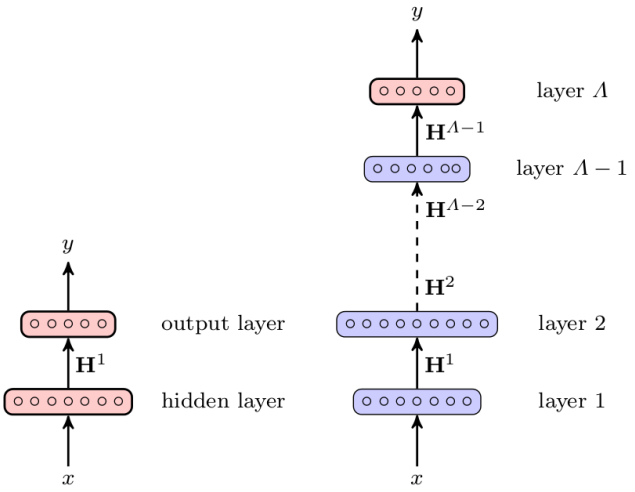


Fig. 3 left: A standard MLP with one hidden layer and one output layer. **right:** A Deep Neural Network, with $\Lambda - 1$ model layers (trained in an unsupervised way) and a final decision layer (trained in a supervised way).

$\lambda \in \{1, \ldots, \Lambda\}$. The first layer is fed with the input, while the last layer computes the output of the network $\mathbf{H}^\Lambda$. Let $\{N^1, \ldots, N^\lambda, \ldots, N^\Lambda\}$ denote the number of neurons of each layer. Each layer computes the vector output $\mathbf{H}^\lambda = \{h_i^\lambda\}, \forall \lambda \in \{1, \ldots, \Lambda\}, i \in \{1, \ldots, N^\lambda\}$, where h_i^λ denotes the output of the i^{th} neuron of layer λ, which is computed using the following equation:

$$h_i^\lambda = f^\lambda \left(\sum_{j=0}^{N^{\lambda-1}} \left[w_{i,j}^\lambda \, . \, h_j^{\lambda-1} \right] \right) \quad \forall i \in \{1, \ldots, N^\lambda\} \tag{6}$$

where $w_{i,j}^\lambda \in \mathbb{R}$ are the weights, and f^λ is a non-linear activation function over the weighted sum: typically a sigmoid function for the $\Lambda - 1$ first layers, and a softmax function for the last layer, so that the outputs sum to one and can approximate *a posteriori* probabilities. Note that $w_{i,0}^\lambda$ is the bias associated to the i^{th} neuron of layer λ through a fictitious input $h_0^{\lambda-1} = 1$.

The deep network computes iteratively from layer 1 to layer λ the equation 6 that can be rewritten at the layer level using matrices as:

$$\mathbf{H}^\lambda = f^\lambda(\, \mathbf{W}^\lambda \, . \, \mathbf{H}^{\lambda-1} \,) \quad \forall \lambda \in \{1, \ldots, \Lambda\} \ , \tag{7}$$

where $\mathbf{W}^\lambda = \{w_{i,j}^\lambda\} \in \mathbb{R}^{N^\lambda \times N^{\lambda-1}}$ is the matrix of parameters to be learnt.

Note that the model layers and the decision layers compute the same quantities, they only differ by the way they are learnt. Let us now described the two-stage training procedure of the DNN.

3.1.2 Training the model layers

The auto-encoder framework allows to learn non linear feature extractors in an unsupervised way. It learns an encoder function e, that maps an input x to a hidden representation $e(x)$, and a decoder function d, that maps back e to the input space as $\hat{x} = d(e(x))$. The estimation $\hat{x}$ is the reconstruction of x through the layer. The encoder and decoder parameters are learnt to minimize a reconstruction error on the training set, similarly to the training of an MLP using back-propagation of the error.

Once trained, the encoder function e is stacked as a new model layer in the deep architecture, while the decoder function d is eliminated from the network. The output of the new stacked layer is fed at the input of a new auto-encoder of the next new layer. Many layers can be stacked, like in Fig. (4), depending on the complexity of the input space.

3.1.3 Training the DNN-HMM combination

The DNN and HMM models have to be learnt so that the DNN provides the emission probabilities to the HMM. Training a neural network-HMM combination can be performed either using joint-training [27,28,22], or independently. Joint training iteratively refines the objectives of the DNN during the training of the HMM, as EM algorithm does for GMM. The independent training scheme first trains a standard HMM-GMM model, and then trains the neural network using the frame label derived from a Viterbi forced alignment on the character sequence ground truth using the trained HMM-GMM character models. In any case, the *a posteriori* probabilities are divided by the prior state probabilities in order to be rescaled into likelihoods. Note that some variants have also been proposed such as the tandem training [40,41], where the outputs of a neural network are used as features on which a standard GMM-HMM is learnt. Training a DNN requires many more iterations than training a HMM, therefore in this work the two models have been trained independently. Once the DNN and HMM have been trained, a few joint-training iterations have been performed, but no significant improvement was observed.

4 Word-spotting system overview

We now describe the whole keyword spotting system. Its input is a document image, in which the set of keywords is spotted, while the output is the location hypotheses of each keyword in the document. It has been implemented using various HMM architectures: a standard GMM-HMM system, an hybrid MLP-HMM, and the DNN-HMM. In the following sections we will provide the details of the processing steps: line segmentation, pre-processing, feature extraction and training of the models.

4.1 Line segmentation

The line segmentation process is an important and difficult task, mainly due to variable skew angle along the document or even along the text line, and adjacent text lines. We have implemented a line segmentation method based on a connected component analysis (see [8] for details).

4.2 Pre-processing

Pre-processing is applied to text line images in order to eliminate noise and to ease the feature extraction procedure. In an ideal handwriting model, the words are supposed to be written horizontally and with ascenders and descenders aligned along the vertical direction. In real data, such conditions are rarely respected. We use skew and slant correction so as to normalize the text line image [42] (See Figures 5 and 6). A contour smoothing is then applied to eliminate small blobs on the contour.

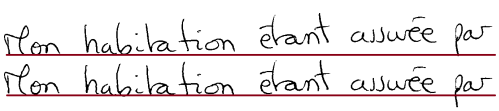


Fig. 5 Deskew correction

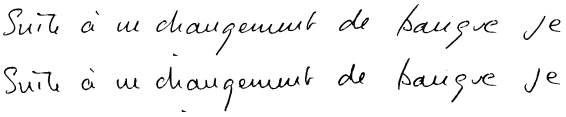


Fig. 6 Deslant correction

4.3 Feature extraction

Each text line image is divided into vertical overlapping windows called frames. For the DNN based system, the pixels of the frames are directly taken as inputs, whereas, for GMM-HMM and MLP-HMM based systems, a feature vector is computed for each frame.

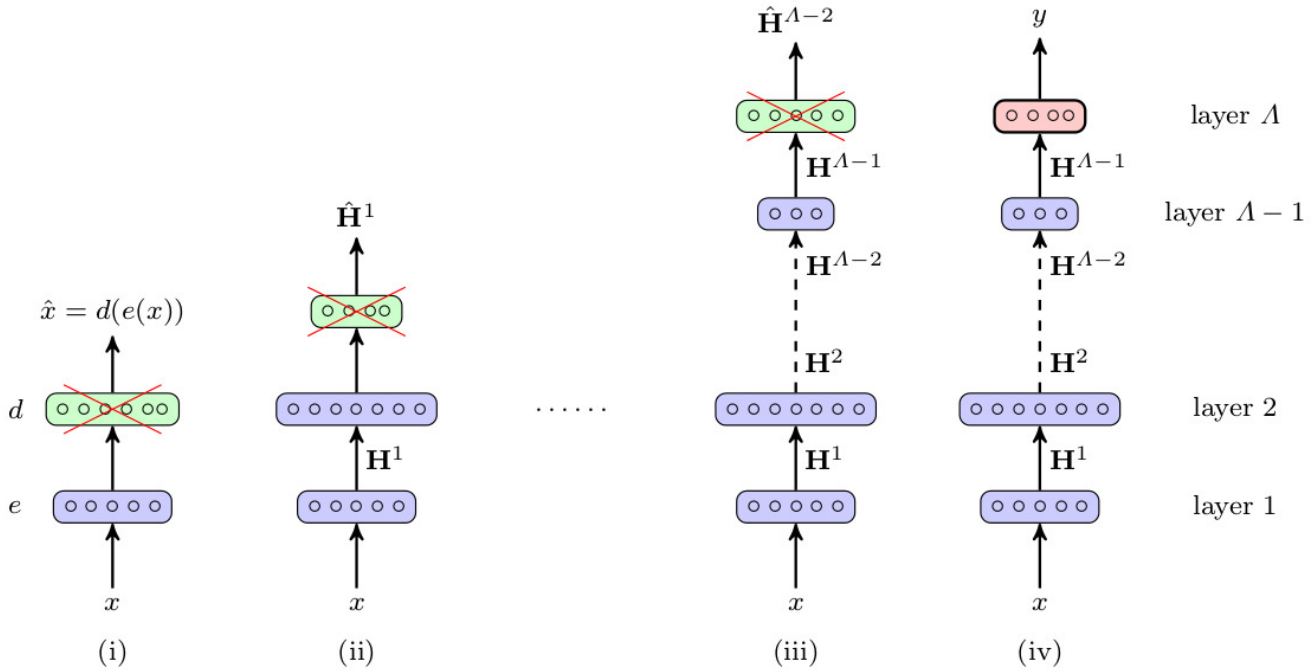


Fig. 4 (i) An auto-associator (e, d) is learnt on the input $\mathbf{x}$. The encoder function e has been stacked to produce a hidden representation $\mathbf{H}^1$ of the data, while d is dropped. (ii) A new auto-associator is then learnt on the hidden units $\mathbf{H}^1$ to produce $\hat{\mathbf{H}}^1$. (iii) This process is repeated to get a compressed representation of the data. (iv) At last, a decision layer is added and a standard supervised gradient back-propagation is applied on the whole network.

This is because GMM models cannot cope with high dimensional binary data.

The feature set is inspired by [43], and has shown its efficiency in the 2009 ICDAR word recognition competition [26]. It is based on density and concavity features. It is made of 26 features:

- 9 are baseline independent (for instance: black pixels density in the window and in every column of this window, position of the writing pixels)
- 17 are baseline dependent (for instance: black pixel density upon and under baselines, local pixels configurations regarding baselines, ...)

4.4 Models training

In this study, an independent training of the HMM and the DNN has been carried out, as discussed in section 3.1.3.

The HMM training has been performed in a traditional way. We have considered $N_c = 71$ characters models: 26 lower case letters, 26 capital letters, 10 digits, a space model, accented letters (é, è, ê, à) and punctuation marks (.,', -, /) in order to model the lines of text. All these character models have been trained with a standard GMM/HMM embedded training at the line level using each text line image generated from the ground truth location of the words in the document. The hyper-parameters of the HMM have been experimentally determined in order to maximize the word recognition rate on a validation dataset. We found that the best combination of parameters was $N_s = 4$ states per character and 5 gaussians per state, and that the best frame window width was $N_p = 8$ pixels with an overlapping factor of $N_r = 5$ pixels.

We now provide the implementation details for training the networks, and especially the DNN architecture. Our implementations were carried out on a standard desktop computer using the *Theano* library [44].

Data preparation: The networks have to be trained so as to provide a posteriori probabilities of each character model state. A frame-labeled training database is therefore needed. It is build using a forced Viterbi alignment of the character sequence groundtruth on the image using a first GMM/HMM trained on the database. This frame labeling of the learning document dataset has led to more than one million frames.

The training of the network requires a huge amount of memory as it needs to store the network weights and the training samples. Therefore, the number of training samples has been limited to approximately 340.000 frames in order to make training affordable on standard desktop computer.

A common training issue with neural networks is that rare classes in the training dataset ('z', 'w', 'k' for

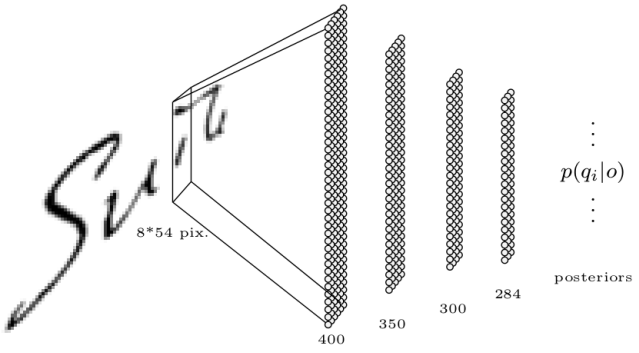


Fig. 7 The DNN-432 architecture. The raw pixels from 54×8 pixels frames are taken directly as inputs. 3 model layers (resp. 400, 350 and 300 neurons) provide a high level representation of the data, while the last layer outputs the posteriors $p(q_i|o)$ for each of the 284 classes.

example for the french language) tend to be excluded from the network output decisions. To avoid this problem, the classes have been balanced to 1.200 samples per class. For classes with less than 1.200 samples, random duplication of samples was performed.

Features: The MLP and DNN have been trained using the same feature set as for GMM (26 features), and using a contextual feature set made of the features from three consecutive frames ($3 \times 26 = 78$ features). The DNN has also been trained directly on pixels values from the whole frame, leading to a 54-pixels height $\times$ 8-pixel width = 432 feature set [3], in order to evaluate its ability to learn high level features from raw data. This configuration is illustrated on Figure 7. These networks are called thereafter **DNN-26**, **DNN-78** and **DNN-432**, **MLP-26** and **MLP-78**. The number of outputs of each network is the number of states of the HMM.

Sizing the networks: Optimizing the DNN architecture is not an easy task due to the lack of theoretical rules to set the hyper-parameters: number of hidden layers, and number of neurons for each hidden layer. The parameters are optimized by maximizing the word recognition rate of the whole system (network + HMM) over the RIMES 2009 word validation dataset (7000 words).

Regarding the choice of the optimal size of the network, several attempts have been made according to some experimental results found in the literature. The number of hidden layers of each DNN has been set to 3 as proposed in [45]. We have also tested deeper architectures with 4 and 5 hidden layers, but the training was much longer, without any significant improvement of the error on the validation dataset. Therefore, we have kept for the remaining experiments a 3-hidden layer network.

The tuning of the number of neurons in the hidden layers is also difficult. In [46], three different strategies have been tested for recognizing handwritten digits from the MNIST database: the three architectures have increasing / constant / decreasing layer sizes from the bottom to the top layers, and the best results were obtained with layers of same size. We did not performed an extensive search, but for our problem the best results were achieved for a decreasing number of neurons (or increasing when the number of outputs is higher than the number of inputs, as it is the case for DNN-26 and DNN-78).

Finally, Table 1 shows the set of optimal hyper-parameters for a three hidden layers network.

Network	N^0 (inputs)	N^1	N^2	N^3	N^4 (outputs)
DNN-26	26	200	225	250	284
DNN-78	78	200	225	250	284
DNN-432	432	400	350	300	284
MLP-26	26	155	$\emptyset$	$\emptyset$	284
MLP-78	78	181	$\emptyset$	$\emptyset$	284

Table 1 Number of neurons N^i for each hidden layer h_i for the three DNN and the two MLP: DNN-26 and MLP-26 are trained on the 26 features set, DNN-78 and MLP78 are trained on 3*26 features set over frames at $t-1$, t and $t+1$, and DNN-432 is trained on the raw pixels. The number of outputs of each network is the number of states of the HMM: 4 states * 71 characters = 284.

Gradient parameters: Deep architectures have also a set of parameters controlling the training phase of the network, namely the weight decay parameter η and the stopping criterion. We classically used a weight decay that decreases with the number of iterations [46] according to the following equation :

$$\eta_n = \frac{\eta_0}{n \times \frac{\eta_0}{10} + 1}$$

where η_n is the weight decay at iteration n, and η_0 is the initial weight decay, set to 0.001.

We have used a common stopping criterion that consists in evaluating the mean square error between the outputs of the network and its target (resp. itself for the model layers). Training is stopped when the error is lower than an ε value.

Concerning the stopping criterion devoted to the model layers, we have observed in our experiments that ε should not be too low in order to limit the overfitting of the model layers. A value of 0.01 has been retained[4].

[3] A vertical scaling procedure is applied to normalize the height of the text line image to $h = 54$.

[4] An alternative to this strategy is to use a small ε, with a maximum number of iterations, as it is the case in [45]

For the fine tuning (final backpropagation applied to the whole network), a very small improving ratio is more suitable, to prevent from stopping the training too early. An $\varepsilon = 0.0001$ has been chosen, coupled with a validation procedure to retain the best network.

5 Experiments and results

In order to compare the performance of our approach, several spotting systems have been evaluated: standard GMM-HMM, hybrid MLP-HMM, and DNN-HMM. These systems have been trained on the public RIMES database. We evaluate our approach on a multiple keywords spotting task.

5.1 RIMES database

The RIMES database includes 1150 french incoming mails (see Figure 8) from different writers [23], as well as 36000 annotated isolated words extracted from these mails. As the coordinates of the words inside the mail images were not available, we have recovered them using a simple pattern matching method. For our experiments, 950 mails have been used for training, 100 documents for validation and 100 for test.

5.2 Experimental protocol

In order to evaluate the keyword spotting system in a database of D documents, we compute recall and precision measures. Different operating points are obtained by balancing the competition between relevant and irrelevant information in the model through the variation of the hyper-parameter G (see Figure 2). It enables to describe a whole recall/precision curve: a value of G close to 1 gives an advantage to the relevant information at the expense of the rejection model and thus will favor recall against precision. On the contrary, values of G close to 0 will favor precision. From an applicative point of view, the user of the system may choose the suitable value of G given its application constraints.

Given a document d, let $N(d)$ be the number of keywords to spot, $N_{ok}(d)$ be the number of correctly spotted keywords in this document, and $N_{fd}(d)$ the number of false detections. Recall and precision means (respectively R_{mean} and P_{mean}) for a whole database containing D documents are computed as follows (equations 8 and 9):

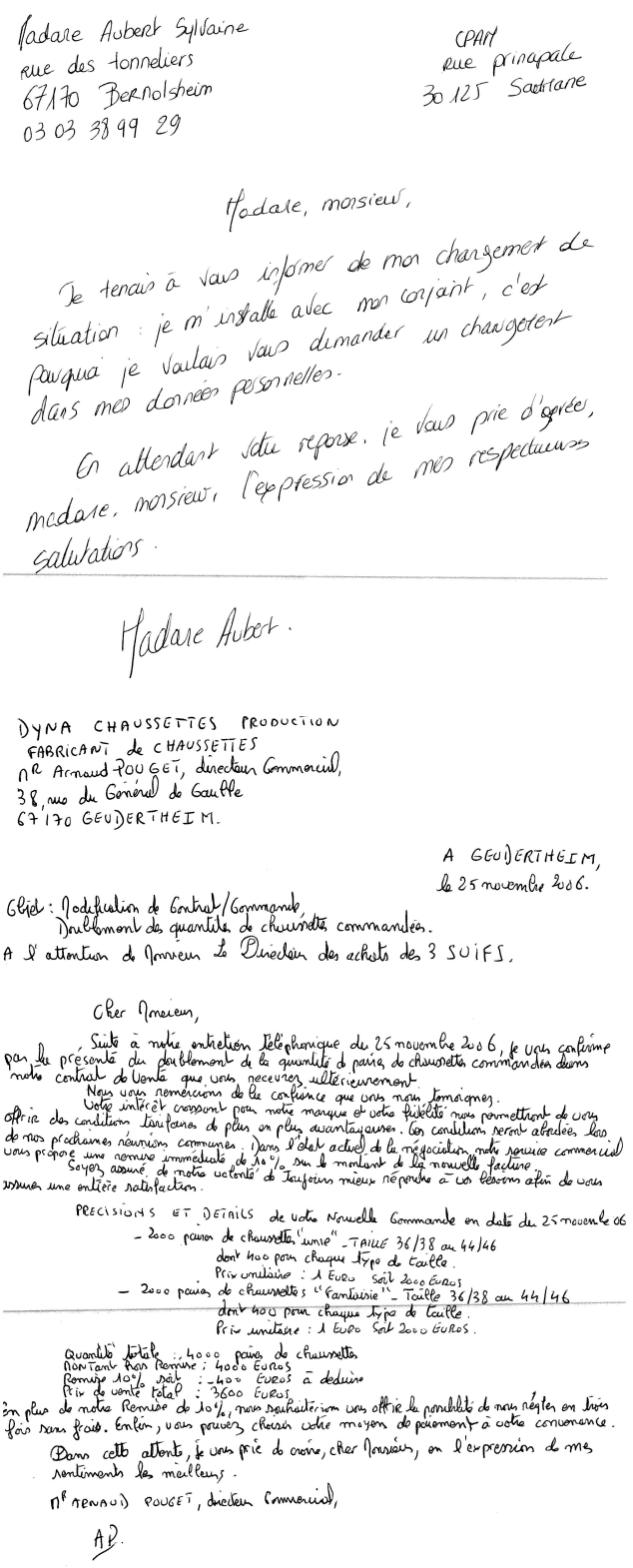


Fig. 8 Incoming mails from RIMES database [23].

$$R_{mean} = \frac{1}{D} \sum_{d} \frac{N_{ok}(d)}{N(d)} \tag{8}$$

$$P_{mean} = \frac{1}{D} \sum_{d} \frac{N_{ok}(d)}{N_{ok}(d) + N_{fd}(d)} \tag{9}$$

Computing statistically significant recalls and precisions requires to have enough keywords to spot in a document database. However, we have observed that the most frequent words do not exceed a few occurrences in the whole database. Therefore, spotting a lexicon of L keywords over the database would lead to a too small $N(d)$ value to correctly estimate recall and precision. For that reason, we have turned towards the use of a document-based protocol.

The experimental protocol is the following: a L-keyword lexicon is generated by picking up randomly 10 words among the words of the document to spot in. Small words with less than 5 characters are discarded. These 10 keywords are completed with $L - 10$ words which are not present in the current document under study.

5.3 Multiple keywords spotting results

We present the spotting performance for three different keyword lexicon sizes: 10, 100 and 500. Figures 9,10 and 11 show the recall-precision results for a lexicon containing respectively 10, 100 and 500 keywords.

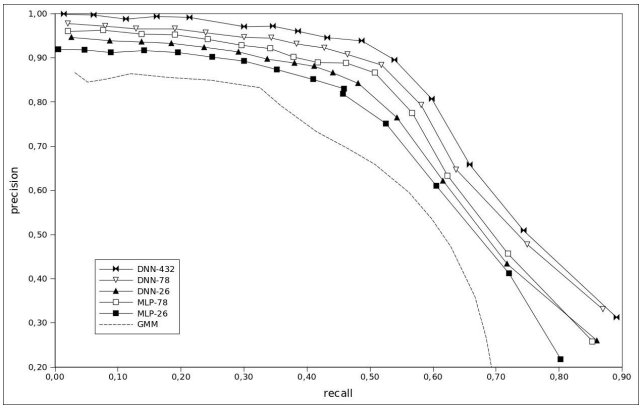


Fig. 9 Recall-precision results for all combinations with a 10-keyword lexicon size.

A first comparison between the different systems show that the deep HMM system clearly outperforms the other classifiers, whatever the value of G. These experiments show that the behavior of the system is the same whatever the lexicon size: the neural based discrimination always provides better results than GMM. The deep HMM system provides the best results if used directly with raw pixels. A more compact presentation of our results is given in Table 2 for each system through the classical break even point evaluation measure corresponding to the operating point where Recall = Precision. The first column (lexicon size = 1) shows that

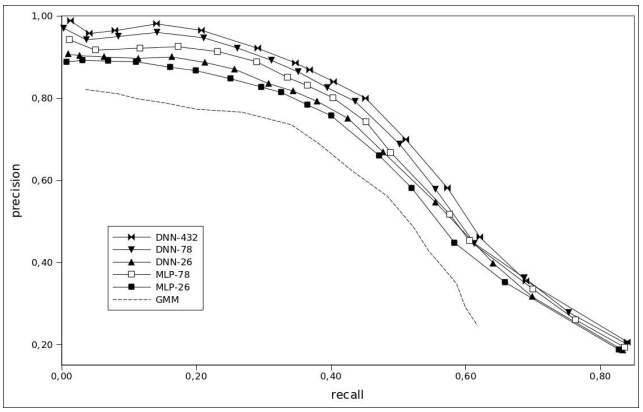


Fig. 10 Recall-precision results for all combinations with a 100-keyword lexicon size.

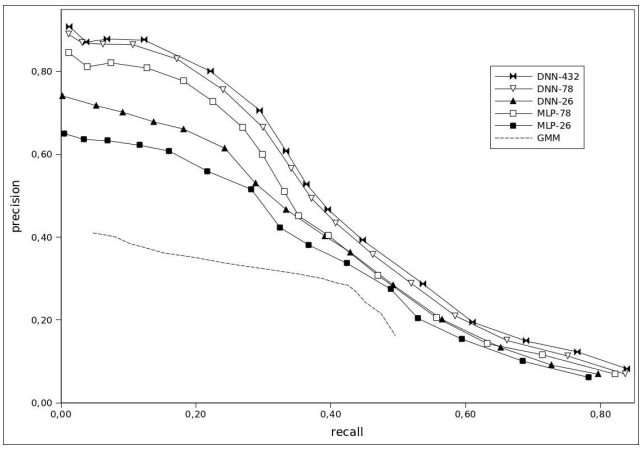


Fig. 11 Recall-precision results for all combinations with a 500-keyword lexicon size.

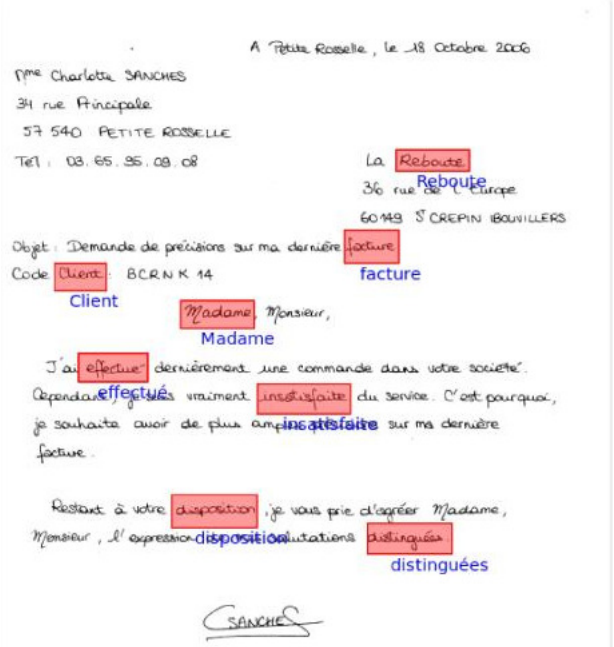
as expected, spotting a single word leads to better retrieval results.

Lexicon size	1	10	100	500
GMM	68.7	57,5	50,9	28,2
MLP-26	70.4	60,6	53,9	37,6
DNN-26	71.9	61,8	55,2	39,7
MLP-78	54.6	62,7	55,5	39,9
DNN-78	74.0	64,1	56,3	41,0
DNN-432	75.8	65,8	57,5	42,4

Table 2 Break even point for different keyword lexicon sizes, for all the systems.

Note that a simple MLP used with some context (previous and next frames) also performs well. As a conclusion, one can say that the deep architecture is not only able to handle high dimensional inputs, but is also able to learn smart discriminating features that provide better results than a dedicated feature set. An

expected result is the behavior of the system when the lexicon size increases: the system performance decrease as the lexicon size increases since more false alarms are generated. A more remarkable result is that the system performance don't fall down when using a 500-word lexicon, allowing an efficient content-based categorization of document images, as shown on [8].



Keyword	found/present
Reboute	1/1
insatisfaite	1/1
effectué	1/1
facture	1/2
client	1/1
Madame	1/2
distinguées	1/1
disposition	1/1
lampadaire	0/0
novembre	0/0

Fig. 12 Example of a multiple keywords spotting result. The lexicon is listed on the lower part of the figure, with two words that do not appear in the document ("lampadaire" and "novembre"). No confusion is made but one utterance of each word, *Madame* and *facture*, is missed. In this example, recall is $6/8 = 75\%$ and precision is $8/8 = 100\%$

6 Conclusion

A multiple keywords spotting system in handwritten unconstrained documents has been introduced. It relies on a global modelling based on HMMs allowing a dual representation of the relevant and the irrelevant information. As we model an entire line of text, a global segmentation/recognition/rejection decision is taken at the line level, avoiding the word segmentation problems. In order to strongly discriminate the local information, an efficient deep neural network has been introduced. The decoding stage of the proposed framework is vocabulary-independent, meaning that the keyword lexicon is independent of the training process and can be set given the application needs.

Results on the RIMES database illustrate the power of the combination scheme. The best results are obtained with the deep architecture trained on raw data, which demonstrates the ability of this architecture to automatically infer high level representation of handwriting.

Future works might investigate alternative learning strategies, such as HMM discriminative learning or Conditional Random Fields (CRF), to propose a fully discriminant method, from the input data to the sequence label. Knowing the language of the documents, possible improvement might also result in the use of language models in order to enhance the performance. Regarding the target application, the proposed approach could also be extended to the search for regular expressions including joker characters '?' and '$\star$', for example by searching for the sequence "canc$\star$" in order to spot any of the words "cancel", "cancellation" or "cancelled". We believe it could be of interest for applications such as automatic document categorization or indexation.

References

1. H. Cao and V. Govindaraju, "Template-free word spotting in low-quality manuscripts," *ICDAR*, pp. 392–396, 2007.
2. C. Choisy, "Dynamic handwritten keyword spotting based on the nshp-hmm," *Proceedings of the Ninth ICDAR*, vol. 1, pp. 242–246, 2007.
3. J. A. Rodríguez-Serrano, F. Perronnin, and J. Llados, "A similarity measure between vector sequences with application to handwritten word image retrieval," *CVPR*, pp. 1722–1729, 2009.
4. J. A. Rodríguez-Serrano and F. Perronnin, "Handwritten word-spotting using hidden markov models and universal vocabularies," *Pattern Recognition*, pp. 2106–2116, 2009.
5. C. Chatelain, L. Heutte, and T. Paquet, "Recognition-based vs syntax-directed models for numerical field extraction in handwritten documents," *ICFHR, Montreal, Canada*, p. 6p, 2008.
6. V. Frinken, A. Fischer, R. Manmatha, and H. Bunke, "A novel word spotting method based on recurrent neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 2, pp. 211–224, 2012.
7. A. Fischer, A. Keller, V. Frinken, and H. Bunke, "Lexicon-free handwritten word spotting using character hmms," *Pattern Recognition Letters*, vol. 33, no. 7, pp. 934 – 942, 2012.

8. T. Paquet, L. Heutte, G. Koch, and C. Chatelain, "A categorization system for handwritten documents," *International Journal on Document Analysis and Recognition*, vol. 15, no. 4, pp. 315–330, 2012.
9. A. Vinciarelli, S. Bengio, and H. Bunke, "Offline recognition of unconstrained handwritten texts using hmms and statistical langage models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 709–720, 2004.
10. S.Thomas, C.Chatelain, L.Heutte, and T.Paquet, "An information extraction model for unconstrained handwritten documents," *ICPR, Istanbul, Turkey*, p. 4, 2010.
11. T. Rath and R. Manmatha, "Features for word spotting in historical manuscripts," *ICDAR*, pp. 218–222, 2003.
12. T.Adamek, N.Connor, and A. Smeaton, "Word matching using single closed contours for indexing handwritten historical documents," *International Journal on Document Analysis and Recognition*, vol. 9, no. 2, pp. 153–165, 2007.
13. M.Rusiñol, D.Aldavert, R.Toledo, and J.Lladós, "Browsing heterogeneous document collections by a segmentation-free word spotting method," *ICDAR*, pp. 63–67, 2011.
14. A. Fischer, A. Keller, V. Frinken, and H. Bunke, "Hmm-based word spotting in handwritten documents using subword models," *ICPR*, pp. 3416–3419, 2010.
15. P. Woodland and D. Povey, "Large scale discriminative training of hidden markov models for speech recognition," *Computer Speech and Language*, vol. 16, no. 1, pp. 25 – 47, 2002.
16. T.M.T.Do and T.Artières, "Maximum margin training of gaussian hmms for handwriting recognition," *ICDAR*, pp. 976–980, 2009.
17. J.Keshet, D.Grangier, and S.Bengio, "Discriminative keyword spotting," *Speech Communication*, vol. 51, pp. 317–329, Apr. 2009.
18. A.Ganapathiraju, J.Hamaker, and J.Picone, "Hybrid svm/hmm architectures for speech recognition," *Interspeech*, pp. 504–507, 2000.
19. B.Q.Huang, C. J. Du, Y. B. Zhang, and M. T. Kechadi, "A hybrid hmm-svm method for online handwriting symbol recognition," *IEEE International Conference on Intelligent Systems Design and Applications*, pp. 887–891, 2006.
20. S. Boquera, M. Bleda, J.Gorbe-Moya, and F. Zamora-Martínez, "Improving offline handwritten text recognition with hybrid hmm/ann models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 4, pp. 767–779, 2011.
21. A.Graves, M.Liwicki, S.Fernandez, R.Bertolami, H.Bunke, and J.Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 855–868, 2009.
22. S. Marukatat, T. Artières, P. Gallinari, and B. Dorizzi, "Sentence recognition through hybrid neuro-markovian modeling," *ICDAR*, pp. 731–735, 2001.
23. E. Grosicki and H. El-Abed, "Icdar 2009 handwriting recognition competition," *ICDAR*, pp. 1398–1402, 2009.
24. M. A. El-Yacoubi, M. Gilloux, and J.-M. Bertille, "A statistical approach for phrase location and recognition within a text line: An application to street name recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 2, pp. 172–188, 2002.
25. L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Readings in speech recognition*, pp. 267–296, 1990.
26. Y. Kessentini, T. Paquet, and A. Benhamadou, "Off-line handwritten word recognition using multi-stream hidden markov models.," *Pattern Recognition Letters*, vol. 31, pp. 60–70, 2010.
27. Y. Bengio, Y. LeCun, C. Nohl, and C. Burges, "Lerec: a nn/hmm hybrid for on-line handwriting recognition," *Neural Computation*, vol. 7, pp. 1289–1303, 1995.
28. S. Knerr and E. Augustin, "A neural network-hidden markov model hybrid for cursive word recognition," *ICPR*, vol. 2, pp. 1518–1520, 1998.
29. A. Ganapathiraju, J. Hamaker, and J. Picone, "Hybrid svm/hmm architectures for speech recognition," *Speech Transcription Workshop*, pp. 504–507, 2000.
30. A. Mohamed, G. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 14–22, 2012.
31. G.E.Dahl, D.Yu, L.Deng, and A.Acero, "Context-dependent pre-trained deep neural networks for large vocabulary speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 30–42, 2012.
32. S. Ortmanns, H. Ney, and X. Aubert, "A word graph algorithm for large vocabulary continuous speech recognition," *Computer Speech Language*, vol. 11, no. 1, pp. 43 – 72, 1997.
33. G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.
34. D.C.Ciresan, U.Meier, L.M.Gambardella, and J.Schmidhuber, "Convolutional neural network committees for handwritten character classification," *ICDAR*, pp. 1135–1139, 2011.
35. X. Niu and C. Suen, "A novel hybrid cnnsvm classifier for recognizing handwritten digits," *Pattern Recognition*, vol. 45, no. 4, pp. 1318 – 1325, 2012.
36. H. Lee, P. Pham, Y.Largman, and A. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," *NIPS*, pp. 1096–1104, 2009.
37. Q. Le, W. Zou, S. Yeung, and A. Ng, "Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis," *CVPR*, pp. 3361–3368, 2011.
38. Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," *NIPS*, pp. 153–160, 2007.
39. M. Ranzato, Y.-L. Boureau, and Y. LeCun, "Sparse feature learning for deep belief networks," in *NIPS*, 2007.
40. J. Schenk and G. Rigoll, "Novel Hybrid NN/HMM Modelling Techniques for On-line Handwriting Recognition," *IWFHR*, 2006.
41. P. Dreuw, P. Doetsch, C. Plahl, and H. Ney, "Hierarchical hybrid MLP/HMM or rather MLP features for a discriminatively trained gaussian HMM: a comparison for offline handwriting recognition," *IEEE International Conference on Image Processing*, 2011.
42. F. Kimura, S. Tsuruoka, Y. Miyake, and M. Shridhar, "A lexicon directed algorithm for recognition of unconstrained handwritten words," *IEICE Transactions on Information & Syst.*, vol. E77-D, no. 7, pp. 785–793, 1994.
43. R. Al-Hajj, C. Mokbel, and L. Likforman-Sulem, "Combination of hmm-based classifiers for the recognition of arabic handwritten words," *ICDAR*, pp. 959–963, 2007.
44. J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, "Theano: a CPU and GPU math expression compiler," June 2010.
45. Y. Bengio, "Learning deep architectures for ai," *Found. Trends Mach. Learn.*, vol. 2, pp. 1–127, 2009.
46. H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin, "Exploring strategies for training deep neural networks," *J. Mach. Learn. Res.*, vol. 10, pp. 1–40, June 2009.