Pattern Recognition 88 (2019) 312–320

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Word spotting and recognition via a joint deep embedding of image and text

Mohamed Mhiri*, Christian Desrosiers, Mohamed Cheriet

École de technologie supérieure, Montreal (QC), H3C 1K3, Canada

ARTICLE INFO

Article history:
Received 15 February 2018
Revised 1 November 2018
Accepted 17 November 2018
Available online 20 November 2018

Keywords:
Representation learning
Word image representation
Word text representation
Word spotting
Word recognition

ABSTRACT

This work addresses three important yet challenging problems of handwritten text understanding: word recognition, query-by-example (QBE) word spotting and query-by-string (QBS) word spotting. In most existing approaches, these related tasks are considered independently. We propose a single unified framework based on deep learning to solve all three tasks efficiently and simultaneously. In this framework, an end-to-end deep neural network architecture is used for the joint embedding of handwritten word texts and images. Word images are embedded via a convolution neural network (CNN), which is trained to predict a representation modeling character-level information. The output of the last convolutional layer is considered as representation in the joint embedding subspace. Likewise, a recurrent neural network (RNN) is used to map a sequence of characters to the joint subspace representation. Finally, a model based on multi-layer perceptrons is proposed to predict the matching probability between two embedding vectors. Experiments on five databases of documents written in three languages show our method to yield state-of-the-art performance for QBE and QBS word spotting. The proposed method also obtains competitive results for word recognition, when compared against approaches tailored specifically for this task.

© 2018 Elsevier Ltd. All rights reserved.

1. Introduction

Understanding handwritten text in document images is an essential problem that receives a growing amount of attention from the pattern recognition community. This problem involves various challenging tasks including *word recognition*, where the goal is to identify the word corresponding to a given region of the document image, and *word spotting*, which aims at finding all occurrences of a query word in a dataset of document images [16,33,40,44]. Word spotting can further be divided in two different scenarios: *query-by-example* (QBE), for which the query word is an image, and *query-by-string* (QBS), where the query is a text string. Despite significant research efforts, handwritten word recognition and spotting remain challenging problems due to several factors, including the large intra-class variability in handwriting shapes and the poor quality of handwritten manuscripts [5,20,25].

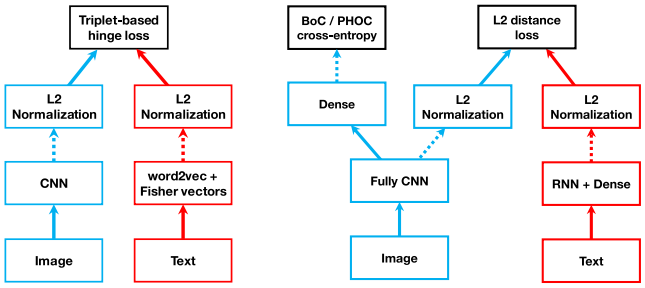
Following previous studies such as [2], we suppose in this work that document images have been segmented into individual words, which serve as candidates for matching the query word. Localizing and segmenting words in images are well-studied problems, for which many effective methods exist [7,34,41]. Moreover, we assume for word recognition that a dictionary (or lexicon) is provided at test time, and that words are recognized by matching them against entries in the dictionary. This problem setting, also used in previous works like [2], limits the recognition to dictionary words. However, as shown in our experiments, it also has the important benefit of allowing cross-database / language transfer of learned information, using a different lexicon in testing than in learning.

Although word spotting and recognition are closely related, most existing approaches consider these tasks separately [18,44]. One of the main reasons for this is that the text and image of words are not directly comparable. A common strategy for addressing this problem is to learn a common representation space for word texts and images. In [2], Almazan et al. encode word images as Fisher vectors of SIFT descriptors and word texts as Pyramidal Histograms of Characters (PHOC), and use Canonical Correlation Analysis (CCA) to embed these encodings into a common subspace. Words are then recognized or retrieved via a simple nearest-neighbor search in this subspace. An important limitation of this approach, however, is that the representation is based on handcrafted features, instead of being learned directly from the data.

In [54], Wang et al. propose a data-driven approach based on deep neural networks for the joint embedding of image and text sentences. This approach, shown in Fig. 1 (a), converts individ-

* Corresponding author.
E-mail address: mohamed.mhiri.1@ens.etsmtl.ca (M. Mhiri).

https://doi.org/10.1016/j.patcog.2018.11.017
0031-3203/© 2018 Elsevier Ltd. All rights reserved.



a) Embedding model of [54] b) Proposed architecture

Fig. 1. A comparison of the image-text joint embedding approach in [54] and the proposed architecture. Our architecture models character-level information about word images via a bag-of-characters (BoC) or PHOC representation loss, and word texts using a recurrent neural network (RNN).

ual words in a sentence to word2vec vectors [38] which are then combined into a single Fisher vector. On the other hand, images are mapped to a vector using a deep convolution neural network (CNN). The proximity of embedded texts and images is enforced via a triplet-based hinge loss. While the word image representation is learned by the CNN, a fixed representation (i.e., word2vec vectors) is assumed for text labels. A problem with this representation is that it mainly captures the semantic of words, not their structure, and is thus poorly suited for word spotting and recognition.

In this work, we propose an end-to-end deep neural network architecture for the joint embedding of handwritten word texts and images. This architecture, shown in Fig. 1 (b), focuses on representing the structure of word images and texts at the level of characters. Toward this goal, we use an intermediate loss function in the network branch embedding word images (blue color in the figure), which maps these images to a bag-of-characters (BoC) or PHOC representation. Moreover, a recurrent neural network (RNN) followed by a dense layer is used in the text embedding branch (red color in the figure) to map a sequence of characters to the common subspace representation.

The main contributions of this work are as follows:

- An end-to-end method is proposed for the joint embedding of handwritten word texts and images. This method encodes character-level information in word images via a CNN trained with a bag-of-characters (BoC) or PHOC loss function, and in word text using an RNN. Unlike approaches in [2] and [54], our method does not rely on handcrafted features and provides a fully-learned representation.
- Although the learned representation can be used directly to recognize words or retrieve words from a text or image query, we propose a robust matching model to improve the accuracy on these tasks. This model can be employed efficiently to filter out false matches obtained with a standard nearest-neighbor approach. The matching is performed between word images and a pre-defined vocabulary for the recognition task, and between word images or texts for QBE or QBS word spotting, respectively.
- In an extensive experimental setup, involving five databases of documents written in three languages, we show that the proposed method can generalize across different databases. This demonstrates our method's robustness to the high variability of handwriting data as well to languages.

The rest of the paper is organized as follows. Section 2 gives an overview of related work on word spotting and word recognition. In Section 3, we then describe the proposed architecture, detailing the word text and image embedding strategies separately. We also present the matching model employed to refine matches obtained using a nearest-neighbor technique. In Section 4, we evaluate the usefulness of our approach on the word recognition and word spotting tasks, and show its advantages compared to the state-of-the-art. Finally, Section 5 concludes the paper with a summary of main contributions and results.

2. Related work

Over the years, various representations have been proposed for encoding word images, including those based on dictionary learning [36], Scale-Invariant Feature Transform (SIFT) descriptors [27,47], Histogram of Oriented Gradients (HOG) descriptors, and Fisher Vectors [1,2]. However, recent works have shown the advantages of inferring the representation in a data-driven manner using deep learning [3,4]. In [24], a CNN is employed to detect and recognize words in natural images. The work in [43] adapts this approach for the recognition of handwritten words. Instead of using the word class as output to the CNN, the target output word is encoded as a hierarchical bag of uni- / bi- / tri-grams called Pyramidal Histogram of Characters (PHOC) [2]. At the l-th level of this hierarchy, the target word is split into l even-sized parts, each of which are represented as a binary vector of N-gram presence.

In [49], Sudholt et al. present a word spotting approach called PHOCNet, which also maps word images to a PHOC representation using a CNN. In a following work [50], they modified their PHOCNet architecture to process arbitrarily-sized images by adding a spatial pooling layer [22] after the last convolutional layer. In [51], this pooling layer is replaced by a Temporal Pyramid Pooling (TPP) layer, which sub-divides recursively the feature maps along the horizontal axis and aggregates the pooled sub-regions in a single vector. A limitation of this model is that it requires words in document images to be pre-segmented. To overcome this limitation, Wilkinson et al. [55] proposed a model named Ctrl-F-Net for segmentation-free QBS word spotting. In this model, the feature maps obtained from a pre-trained CNN are fed to a region proposal network (RPN) to regress bounding boxes of possible regions of words. A second CNN model is then employed to represent the selected regions in the space of PHOC representations, where the matching is done.

The works mentioned above focus on either word recognition or word spotting, but not both tasks. As presented in the introduction, recent studies have shown the potential of solving these problems together using a joint embedding strategy for word texts and images [2,54]. Inspired by this, Krishnan et al. [28] proposed using a pre-trained CNN called HWNet [29] and linear SVM classifiers to map word images to the same PHOC representation as corresponding text labels. Based on the work of Almazan et al. [2], a common subspace regression (CSR) technique is used to maximize the correlation between text and image vectors in the representation space. Unlike this approach, our proposed method does not assume a fixed representation, and instead employs an RNN to find an optimal encoding for text labels. In [30], Gomez et al. use CNNs to embed both word images and texts (modeled as a one-hot matrix) in a common representation space. A siamese network is trained to predict the Levenshtein edit distance [31] of word pairs [31] from the Euclidean distance between their embedded images or texts. While promising, this approach is only used for QBS word spotting.

Sequential models like Hidden Markov Model (HMM) are often applied to word representations for recognition or matching. Such models typically treat words as a concatenation of many compound HMMs, each one representing single characters [39,45,53]. Several works have focused on extending standard HMM-based models to add contextual information, for instance, by considering surrounding characters [13,15]. However, such strategies may lead

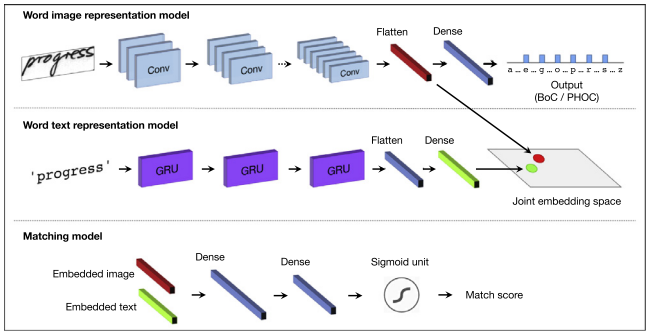


Fig. 2. The proposed framework that learns a joint embedding of word images and texts to address the word spotting and recognition tasks.

to a high number of HMMs, each one with many parameters to learn and, thus, they are not suitable when limited training data is available.

Recurrent neural networks (RNNs) like Bidirectional Long Short-Term Memory (BLSTM) address the problem of limited context by using a specific architecture called memory block, which can preserve contextual information over a long period of time. In BLSTMs, a special type of layer called Connectionist Temporal Classification (CTC) [17,18,56] is also used to map an input sequence to a target label sequence without requiring their alignment. This layer is trained to predict the probability of a given label sequence as a sum over all possible alignments. To the best of our knowledge, the joint embedding method proposed in this current work is the first to combine RNNs and CNNs for solving word recognition and word spotting together.

3. Methodology

The proposed network architecture is illustrated in Fig. 2. In this network, the word spotting and recognition tasks are addressed together by learning a joint embedding of word images and texts. As in recent image representation approaches, a CNN is used for mapping word images to a set of high-level features that are robust to various transformations like small translations [4,12]. In another branch, word texts are embedded to the same subspace via an RNN followed by a dense layer. The RNN is based on Gated Recurrent Units (GRU) layers [11], which learn the dependencies between characters in a word. GRU layers are typically employed to build word image recognition models. In this work, we use these layers in the opposite direction for mapping text to word image representations.

Once word images and texts are embedded in a common subspace, the Euclidean distance can be used to measure the similarity between representations. For QBE word spotting, a query word image is first embedded using the trained CNN, and the framework then returns its nearest word images in the embedding subspace. Likewise, QBS word spotting is carried out by projecting a query string into the embedding subspace. Using a similar idea, the recognition of a query word image is performed by finding the lexicon word whose embedding is nearest. While this nearest-neighbor strategy is highly efficient, it may result in false matches that affect accuracy. To overcome this problem, a more powerful matching model based on a densely-connected neural network is applied in cascade to select the final matches from the list of nearest-neighbors.

The following sections give a formulation of the joint embedding problem, and present the models employed for embedding or matching word images and texts.

3.1. Problem formulation

Let $\mathcal{X} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$ be the training set of N words, each training word i represented by an image $\mathbf{x}_i$ and text $\mathbf{y}_i$. Our goal is to learn an image embedding function f_I and text embedding function f_T such that $\mathbf{x}_i$ and $\mathbf{y}_i$ will be near each other in the embedding subspace. However, simply minimizing the embedding distance between word image-text pairs can lead to the degenerate solution where all images and texts are mapped to same point. A common strategy to avoid this problem uses a contrastive loss [21] that also penalizes non-matching image-text pairs whose embedding distance is below a given threshold. Alternatively, one can use a ranking loss [54] which enforces matching pairs to be closer in the embedding subspace than non-matching ones. An important drawback of these strategies is the necessity to sample a large number of triplets, making the training process more complicated. Moreover, because they only consider pairwise similarity values, these strategies might not preserve the structural information of objects to embed (e.g., the characters in a word).

In this work, we propose an efficient embedding approach which does not require sampling triplets and can preserve the internal structure of words. Denote as $\boldsymbol{\theta}_I$ and $\boldsymbol{\theta}_T$ the parameters of embedding functions f_I and f_T, respectively. We define the embedding problem as minimizing loss function

$$J(\boldsymbol{\theta}_I, \boldsymbol{\theta}_T; \mathcal{X}) = \sum_{i=1}^{N} J_{\text{dist}}\big(f_I(\mathbf{x}_i; \boldsymbol{\theta}_I),\ f_T(\mathbf{y}_i; \boldsymbol{\theta}_T)\big) + J_{\text{struct}}\big(f_I(\mathbf{x}_i; \boldsymbol{\theta}_I),\ \mathbf{s}_i\big), \quad (1)$$

where J_{dist} encodes the embedding distance between a word image $\mathbf{x}_i$ and its corresponding text $\mathbf{y}_i$, and J_{struct} compares $\mathbf{x}_i$ with a fixed-length vector $\mathbf{s}_i$ encoding structure.

In what follows, we drop the word index i and use $\mathbf{f}_I, \mathbf{f}_T \in \mathbb{R}^M$ as shorthand notation for $f_I(\mathbf{x}_i; \boldsymbol{\theta}_I)$ and $f_T(\mathbf{y}_i; \boldsymbol{\theta}_T)$, where M is the size of embedding subspace. Since our word spotting and recognition framework is based on Euclidean distance, we define J_{dist} via the L2 norm:

$$J_{\text{dist}}\big(\mathbf{f}_I, \mathbf{f}_T\big) = \frac{1}{M} \big\| \mathbf{f}_I - \mathbf{f}_T \big\|_2^2. \quad (2)$$

Here, the collapse of the embedding to a single point is avoided by the structure-preserving term J_{struct}.

We consider two well-known models for preserving the structural information of embedded words: bag-of-characters (BoC) and PHOC. Let $\mathbf{s} \in \{0, 1\}^K$ be the structural encoding of a word. For BoC, K is the number of characters in the alphabet, and the k-th bit of $\mathbf{s}$ equals 1 if the corresponding alphabet character is in the word, else the bit is 0. We first learn a parametric model, composed of fully-connected layers (last two rows of Table 1), which takes as input the word image embedding and outputs for each element k of $\mathbf{s}$ the probability p_k that this element is one. Then, J_{struct} is defined as the mean cross-entropy between the predicted probabilities and binary values in $\mathbf{s}$:

$$J_{\text{struct}}(\mathbf{f}_I, \mathbf{s}) = -\frac{1}{K} \sum_{k=1}^{K} s_k \log p_k + (1 - s_k) \log(1 - p_k). \quad (3)$$

As described in the following sections, parameters $\boldsymbol{\theta}_I$, $\boldsymbol{\theta}_T$ are learned jointly in an end-to-end fashion.

3.2. Word image embedding

Table 1 details the CNN architecture employed to embed word images. This network is composed of 13 layers in total, with 8 convolutional layers, 3 max-pooling layers and 2 fully-connected layers. All hidden layers use rectified linear units (ReLU) [4]. Moreover, as recommended in [23], batch normalization is applied before each activation layer. After the last convolution layer, feature

Table 1
The CNN architecture for embedding word images of size 40×170.

Layer type	Output shape
Conv2D (w/ padding)	$32 \times 40 \times 170$
Conv2D (w/ padding)	$32 \times 40 \times 170$
Max-pooling	$32 \times 20 \times 85$
Conv2D (w/ padding)	$64 \times 20 \times 85$
Conv2D (w/ padding)	$64 \times 20 \times 85$
Max-pooling	$64 \times 10 \times 42$
Conv2D (w/ padding)	$128 \times 10 \times 42$
Conv2D (w/ padding)	$128 \times 10 \times 42$
Max-pooling	$128 \times 5 \times 21$
Conv2D (w/ padding)	$256 \times 5 \times 21$
Conv2D (w/o padding)	$256 \times 1 \times 17$
Flatten (our word image representation)	**2176**
Fully-connected (ReLU units)	1000
Fully-connected (sigmoid units)	52*

* *Number of characters in the IAM alphabet, using a BoC encoding.*

maps are flattened into a vector of size 2176, corresponding to our word image embedding $\mathbf{f}_I$. Two fully-connected layers are then used to map this vector to the final BoC or PHOC representation. The output layer has K sigmoid units, each one corresponding to a different element of $\mathbf{s}$. Since multiple elements of $\mathbf{s}$ can be equal to 1, the proposed architecture has no softmax layer before the output.

A problem of using this CNN is that input word images may have different sizes. A simple solution to this problem is to resize the input word images to a pre-determined size, however, this may affect the aspect ratio of images (i.e., the ratio between image width and height). To avoid this situation, we apply a more sophisticated technique, presented in Algorithm 1 , which right-

Algorithm 1: The proposed resizing technique.

Input: A word image $\mathbf{I}$ of size $h \times w$.
Output: A resized word image $\mathbf{I}_R$ of size $h_R \times w_R$.

Initialize $\mathbf{I}_R$ to a white image;
if $h \leq h_R$ *and* $w \leq w_R$ **then**
  $\mathbf{I}_R(i,j) := \mathbf{I}(i,j), \quad i = 1,\ldots,h, \;\; j = 1,\ldots,w;$
else
  if $h_R/h \leq w_R/w$ **then**
    $\mathbf{T} := \text{resize}\big(\mathbf{I}, [h_R, w \times h_R/h]\big);$
    $\mathbf{I}_R\big(i,j\big) := \mathbf{T}(i,j), \quad i = 1,\ldots,h_R, \;\; j = 1,\ldots,w \times h_R/h;$
  else
    $\mathbf{I}_R := \text{resize}\big(\mathbf{I}, [h_R, w_R]\big);$
  end
end
return $\mathbf{I}_R$

pads word images with white space until reaching a fixed size of 40×170 pixels. Fig. 3 shows two examples of word images, and the result of resizing these words using the standard approach and the proposed technique. We see that our technique better preserves the aspect ratio of images.

3.3. Word text embedding

To embed the character string of a word, we use the neural network architecture given in Table 2. This architecture is composed of four Gated Recurrent Units (GRU), alternating between forward and backward input sequences, followed by a dense (i.e., fully-connected) layer with ReLU activations. Each GRU performs the following processing

$$\mathbf{z}_t = \sigma_g\big(\mathbf{W}_z\mathbf{x}_t + \mathbf{U}_z\mathbf{h}_{t-1} + \mathbf{b}_z\big) \tag{4}$$

$$\mathbf{r}_t = \sigma_g\big(\mathbf{W}_r\mathbf{x}_t + \mathbf{U}_r\mathbf{h}_{t-1} + \mathbf{b}_r\big) \tag{5}$$

$$\mathbf{h}_t = \mathbf{z}_t \circ \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \circ \sigma_h\big(\mathbf{W}_h\mathbf{x}_t + \mathbf{U}_h(\mathbf{r}_t \circ \mathbf{h}_{t-1}) + \mathbf{b}_h\big) \tag{6}$$

where $\circ$ is the Hadamard product. In this formulation, $\mathbf{x}_t$ is the input vector, $\mathbf{h}_t$ the output vector, $\mathbf{z}_t$ the update gate vector, and $\mathbf{r}_t$ the reset gate vector. Parameters of a unit correspond to weight matrices $\mathbf{W}_z$, $\mathbf{W}_r$, $\mathbf{W}_h$ and bias vectors $\mathbf{b}_z$, $\mathbf{b}_r$, $\mathbf{b}_h$. For the gate activation function σ_g and output activation function σ_h, we use the standard sigmoid and hyperbolic tangent, respectively.

Input vectors $\mathbf{x}_t$ correspond to a one-hot encoding of each character in the string to embed. To have even-length character sequences, we fix the sequence length to a constant $L_{\max} = 24$ and pad smaller strings with a special *void* character (i.e., $\emptyset$). Hence, each input word text is binary matrix of size $(K+1) \times 24$, where $K+1$ is the number of characters in the alphabet plus the void character. For backward GRUs, input character sequences are processed in reverse. This allows learning bidirectional dependencies between characters in a word.

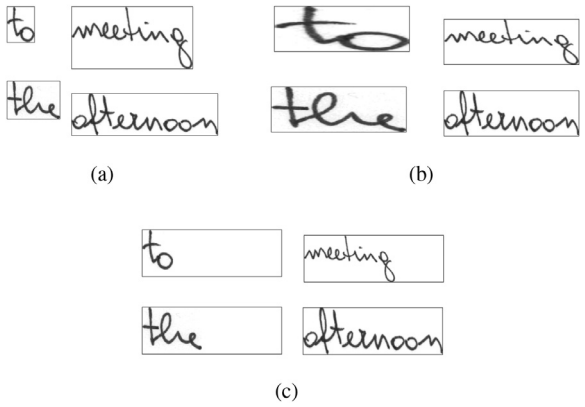


Fig. 3. Word image resizing. (a) Word images of different size and their resized version using (b) the standard resizing technique, and (c) the proposed technique.

Table 2
Architecture of the GRU-based recurrent neural network to embed word texts. Inputs texts are of size $(K+1) \times 24$, where K is the number of characters in the alphabet.

Layer type	Output shape
GRU (forward)	24×64
GRU (backward)	24×64
GRU (forward)	24×64
GRU (backward)	24×64
Flatten	1536
Fully-connected (ReLU, our word text representation)	**2176**

3.4. Multi-layer perceptron matching model

The CNN and RNN embedding models defined in the previous sections are used in conjunction with L2 normalization layers (see Fig. 2) to represent word images and texts in the common learned subspace. The word spotting (QBE or QBS) and word recognition tasks can then be solved as a simple retrieval problem using Euclidean distance. In the case of word spotting, the query word image (QBE) or text (QBS) is embedded with either the CNN or RNN. A nearest-neighbors algorithm is then employed to find the

Table 3
Architecture of the MLP matching model for an input vector of size 4352, corresponding to the concatenation of two embedding vectors.

Layer type	Output shape
Dense (ReLU units)	3000
Dense (ReLU units)	2000
Dense (sigmoid units)	**1**

database word images closest to the query word in the embedding subspace. For word recognition, each word in the lexicon is mapped to the subspace using the RNN. A word image is recognized by finding its nearest lexicon word in the embedding subspace.

In the case of handwritten documents, which have a large shape variability, this simple and efficient strategy may be insufficient to discriminate between words having a similar appearance or spelling. To overcome this problem, we consider the multi-layer perceptron (MLP) model of Table 3 to identify matching words from a list of nearest-neighbors. This MLP, composed of two hidden layers with ReLU activation and a sigmoid unit as output layer, takes as input the concatenated embedding vectors and predicts whether these vectors correspond to the same word.

Training is performed using pairs of embedding vectors, with cross-entropy as loss function. Note that these vectors may correspond to different word representations (text or image), thus providing a rich set of training examples. For instance, the IAM database has around 82K training word images and 11K word texts, giving a total of about 8649M training pairs. However, most of these pairs correspond to non-matching words. To solve this issue, we find for each embedding vector $\mathbf{f}_i$ the $\mathsf{nbNeighbors} = 10$ vectors $\mathbf{f}_j$ nearest to $\mathbf{f}_i$ in the embedding subspace, and use pairs $(\mathbf{f}_i, \mathbf{f}_j)$ as training. The label of a given pair is 1 if $\mathbf{f}_i$ and $\mathbf{f}_j$ correspond to the same word, else the label is 0. This strategy, summarized in Algorithm 2 , is used in order to reproduce the operating conditions of the matching system, which is only applied on the list of nearest-neighbors. Considering nearby embedding vectors also has the benefit of providing challenging examples for training.

Algorithm 2: The training algorithm for the matching model.

Input: A set of L2-normalized embedding vectors $\mathcal{X}$.
Input: Parameters maxEpochs, batchSize and nbNeighbors.
Output: The trained matching model $\mathcal{M}$.

for epoch $= 1, \ldots,$ maxEpochs **do**
  $\mathcal{S} :=$ random subset of batchSize examples from $\mathcal{X}$;
  $\mathcal{T} := \emptyset$;
  foreach $\mathbf{f}_i \in \mathcal{S}$ **do**
    $\mathcal{N}$:=set of nbNeighbors embedding vectors closest to $\mathbf{f}_i$;
    foreach $\mathbf{f}_j \in \mathcal{N}$ **do**
      $\mathcal{T} := \mathcal{T} \cup \{(\mathbf{f}_i, \mathbf{f}_j, \text{label}_{ij})\}$;
    end
  end
  $\mathcal{M} := \text{train}(\mathcal{M}, \mathcal{T})$;
end
return $\mathcal{M}$

4. Experiments

In this section, we evaluate the performance of the proposed method for the word spotting and recognition tasks. Experiments compare our method against state-of-the-art approaches for theses tasks, and assess whether it can generalize across different datasets.

4.1. Databases

Five databases of handwritten documents are used in our experiments: IAM Off-line, RIMES, George Washington (GW), Lord Byron (LB), Institut für Nachrichtentechnik/Ecole Nationale d'Ingénieurs de Tunis (IFN/ENIT). The description of these databases is as follows.

- The Off-line IAM database [35] is a large set of English texts comprised of 115,320 labeled words. In our experiments, we used only a subset of 96,456 labeled word images, remaining ones discarded due to an incorrect segmentation. The pre-defined training and validation sets are combined into a single training set of 82,703 word images corresponding to 10,027 distinct words. The pre-defined test set, with 13,753 word images corresponding to 2904 distinct words, was kept as is. As lexicon, we considered all distinct words from both training and test sets, representing a vocabulary of 11,118 words. Ignoring letter case, words in the IAM database contain $K = 52$ different characters: `!''#&'()*+,-./0123456789:;?abcdefghijklmnopqrstuvwxyz`.
- The RIMES database [19] is a large set of mails written in French. The 2009 version of this database is used in this work, which contains three subsets: training, validation, and test sets. As in the IAM database, all examples from the training and validation sets are combined into a single training set of 51,737 word images corresponding to 4637 distinct words. The test set contains 7464 word images from a set of 1509 distinct words. As lexicon, we considered all distinct words from both training and test sets, representing a vocabulary of 4989 words. Ignoring letter case, words in the RIMES database contain $K = 53$ different characters: `'-/0123456789abcdefghijklmnopqrstuwxyz`°²àâçéêëîïôúû.
- The George Washington (GW) database [46] is a set of English historical documents. It includes 4860 word instances written by two persons, and has a vocabulary of 1124 words. In our experiments, the GW database is tested using a fourfold cross validation approach since there is no official partition in training and test images as in [2].
- The Lord Byron (LB) database [46] is a set of English typewritten texts containing 4988 word instances. This database has a vocabulary of 1569 words.
- The IFN/ENIT database [32] is a set of Arabic texts of Tunisian city and village names written by hundreds of different writers. It is composed of 32,492 images divided into five sets: A, B, C, D, and E. In our experiments, we used set E composed of 4352 images to evaluate our method's ability to transfer knowledge across databases.

4.2. Evaluation protocol

As in [2], we used the character error rate (CER) and the word error rate (WER) to measure word recognition performance. The CER between a query word and the transcribed word is defined as the edit (or *Levenshtein*) distance between these words (i.e., the minimum number of character insertions, deletions, and substitutions needed to transform one word into the other) normalized by

the length of the query word. Likewise, the WER is the percentage of words that are wrongly transcribed.

Word spotting performance is evaluated in terms of mean Average Precision (mAP) and mean Average Precision at K (mAP@K), which are standard measures in retrieval systems. Let AP_q be the average precision for the q^{th} query word image, i.e.,

$$AP_q = \frac{1}{R_q}\sum_{k=1}^{N_q} P(k) \times \text{rel}(k), \tag{7}$$

where N_q is the number of the selected word images that are similar to the query word image, rel(k) is 1 if the rank k is a relevant word image and 0 otherwise, $P(k)$ is the precision at cut-off k in the list, and R_q the number of relevant word images for the query word image. Denoting as Q the total number of queries, the mean Average Precision (mAP) can be defined as

$$\text{mAP} = \frac{1}{Q}\sum_{q=1}^{Q} AP_q. \tag{8}$$

The mean Average Precision at K (mAP@K) extends mAP by considering only relevant words within the top-K predictions [10]. This metric can be expressed as

$$\text{mAP@K} = \frac{1}{Q}\sum_{q=1}^{Q} (\text{AP@K})_q \tag{9}$$

with

$$(\text{AP@K})_q = \frac{1}{\min(R_q, K)}\sum_{k=1}^{K} P(k) \times \text{rel}(k). \tag{10}$$

Once again, we followed the protocol proposed in [2]. For query-by-example (QBE) word spotting, each word image in the test set is used as query. Only queries having relevant word images are used to calculate the mAP. Note that query words are removed from their corresponding ranked list of matches. For query-by-string (QBS) word spotting, each word in the lexicon of the test set is used as query to rank all the testing word images. In the IAM database, the stop-words are not used as queries, however, these words are considered as elements of the database.

4.3. Implementation details

The proposed architecture was developed using the Keras library for deep learning. For training the matching model, we used a maximum of 500 epochs, each one containing a batch of 6000 examples randomly sampled from the training set. The Adam algorithm [26] was employed for parameter optimization. As recommended, the hyper-parameters of this algorithm were kept without any tuning. Training word images were augmented using random rotations ranging from 1 to 10 degrees, random translations with vertical and horizontal shift ranges of 0.05 × height and 0.05 × width, and random zooms with factors between 0.9 and 1.1. Training and testing was carried out on a GeForce GTX 960 GPU with a clock speed of 1190 MHz.

4.4. Results

4.4.1. Query-by-example word spotting

As first experiment, we evaluate the performance of our BoC-based and PHOC-based models on the IAM and GW datasets for the task of QBE word spotting. Table 4 gives the mAP of these two models, using Euclidean distance retrieval or the MLP matching model of Section 3.4, as well as 6 different approaches for this task: 1) simple Fisher Vector (FV) encoding of word images [2], 2) FVs jointly embedded with PHOC word representations and attribute SVMs used for matching [2], 3) PHOCNet [49], 4) the AlexNet architecture pre-trained on the ImageNet database and then fine-tuned using IAM training examples [48], 5) Deep feature embedding using pre-trained CNN [28], and 6) PHOCNet with the Temporal Pyramid Pooling (TPP) layer [51]. Since no test set is provided for the GW dataset, we used four-fold cross validation approach as in [2].

Table 4
Query-by-example (QBE) word spotting performance (mAP) of our BoC-based and PHOC-based models on the IAM and GW databases.

Method	IAM	GW
Fisher vector [2]	15.66%	–
Attribute SVMs [2]	55.73%	93.04%
Deep feature embedding [28]	84.24%	94.41%
PHOCNet [49]	72.51%	92.64%
Fine-tuned CNN [48]	46.53%	–
TPP-PHOCNet (BCA) [51]	84.80%	**97.90%**
Ours (Euclidean dist. + BoC)	72.11%	80.13%
Ours (Euclidean dist. + PHOC)	70.48%	79.55%
Ours (Matching model + BoC)	85.24%	95.22%
Ours (Matching model + PHOC)	**86.37%**	96.31%

Table 5
Query-by-example (QBE) word spotting performance in term of mAP on the GW, LB and IFN/ENIT databases. Our method is trained using training examples of the IAM database.

Method	GW	LB	IFN
Fisher vector [2]	62.72%	86.01%	13.08%
Multi-scale features [37]	66.1%	–	–
Ours (Euclidean dist. + BoC)	**78.60%**	**94.69%**	**28.13%**

It can be seen that our method, both with Euclidean distance retrieval or the MLP matching model, compares favorably with other approaches. Using MLP matching, our PHOC-based model yields a performance near to that of TPP-PHOCNet, considered as the state-of-art for this problem. While TPP-PHOCNet gives a 1.59% higher mAP for the GW dataset, our method offers a 1.57% improvement for the IAM dataset. Results also show a clear advantage of using a trained matching model, with an mAP 13.13% higher than that of Euclidean distance for IAM and 14.91% for GW. Comparing the BoC-based and PHOC-based models, we observe similar performances: the BoC encoding performs better when using Euclidean distance, while PHOC is best for the MLP matching model.

To validate our method's robustness across datasets, we tested the BoC-based model, trained using only IAM examples, on the GW and LB databases which also contain English handwritten documents. For this experiment, all images of the GW or LB databases were considered as queries (i.e., the whole database is used as testing set). Moreover, we measured our method's ability to generalize across different languages by testing it on the IFN/ENIT database whose documents are in Arabic. Results, reported in Table 5, indicate that the learned representation for word images is transferable across datasets, in particular if their documents are in the same language. Comparing against Fisher Vectors, which does not require database-specific training examples, our transferred representation yields mAP improvements of 15.88%, 8.68% and 15.05%, respectively, for the GW, LB and IFN/ENIT databases. With respect to cross-language generalization, we notice an important drop in performance when testing on the IFN/ENIT database, compared to GW and LB which have English documents. However, the mAP obtained by our method trained on English documents (28.13%) is significantly higher than the expected mAP of using a random word ranking (i.e., ∼1.4%). This indicates that the proposed representation captures information that does not depend on language. Our transferred representation also outperforms the approach of

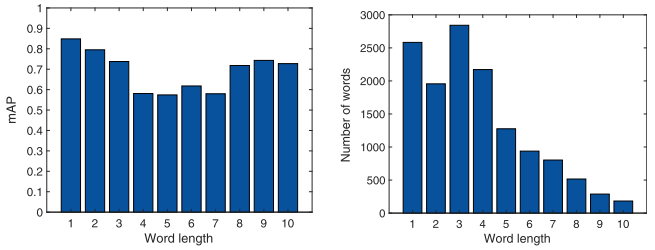


Fig. 4. The mAP of the BoC-based model and occurrence frequency corresponding to different lengths of words in the IAM database.

Table 6
Query-by-string (QBS) word spotting performance in terms of mAP on the IAM database.

Method	IAM test set
Attribute SVMs [2]	73.72%
PHOCNet [49]	82.97%
Deep feature embedding [28]	91.58%
TPP-PHOCNet [51]	**92.97%**
Ours (Euclidean dist. + BoC)	77.32%
Ours (Euclidean dist. + PHOC)	80.23%
Ours (Matching model + BoC)	90.67%
Ours (Matching model + PHOC)	91.88%

our previous work [37], which learns a multi-scale representation in an unsupervised manner using the spherical k-means algorithm.

Fig. 4 analyzes the mAP obtained by our BoC-based model on the IAM testing set, with respect to word length. The distribution of test examples by length is also shown in the figure. We see that our approach performs better with short words than longer ones. This can be explained in part by the heavy-tailed distribution of word lengths, where shorter words are more frequent than longer ones. Conversely, the higher performance for words with over 7 characters, compared to average-length words, could be due to the more characteristic nature of these words (i.e., there are fewer words with similar spelling).

4.4.2. Query-by-string word spotting

Table 6 gives the performance of our method for QBS word spotting on the IAM database. A total of 2904 words from the IAM test set are used as queries, each one represented as a sequence of $L_{\max} = 24$ characters from an alphabet of $K = 52$ possible values plus the void character (see Section 3.3). As in previous experiments, we compared our method against the joint embedding approach of Almazan et al. [2], PHOCNet [49], Deep feature embedding using pre-trained CNN [28], and PHOCNet with the Temporal Pyramid Pooling (TPP) layer [51].

Once again, we observe a higher performance when employing the MLP matching model than Euclidean distance. With MLP matching, the BoC-based model gives an improvement of 7.70% over PHOCNet [49] and of 16.95% over the approach in [2]. Likewise, the PHOC-based model with MLP matching gives an improvement of 8.91% over PHOCNet [49], 18.16% over the approach of [2], and 0.3% over the method of [28]. This highlights the importance of having a fully-learned representation that combines text and image information. While TPP-PHOCNet obtains superior mAP for this dataset, it was designed specifically for word spotting and, unlike our proposed method, cannot be used directly for word recognition.

To illustrate the learned representation, Fig. 5 (left) shows the Euclidean distances between the L2-normalized embedding vectors of five words with similar characters: 'the', 'they', 'there', 'three', 'thousand'. We see that the representation captures character-level information and that the Euclidean distance behaves like the Levenshtein edit distance shown in Fig. 5 (right). For instance, the

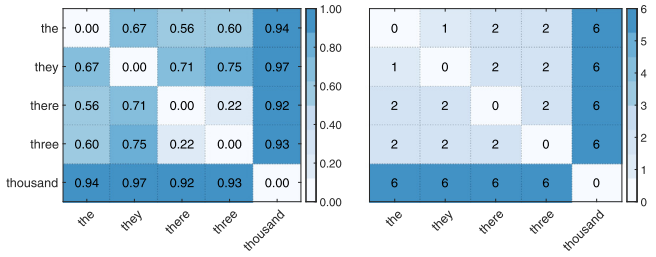


Fig. 5. (**left**) L2 distance between the normalized representations of five words, obtained with the BoC-based model. (**right**) Levenshtein edit distance between same words. Words were selected to have various levels of spelling similarity.

Table 7
Query-by-string (QBS) word spotting performance in terms of mAP@K on the IAM database.

	@5	@10	@15	@20	@25
Euclidean dist. + BoC	81.63%	80.91%	80.50%	80.11%	79.85%
Matching model + BoC	92.36%	92.07%	91.85%	91.69%	91.55%

Table 8
Word and character error rates obtained on test examples of the RIMES and IAM databases and a model trained with BoC representations.

Method	RIMES		IAM	
	WER	CER	WER	CER
HMM [6]	14.7%	–	21.9%	–
HMM [14]	16.8%	–	21.2%	9.1%
CNN-GHMM [8]	**9.2%**	–	20.5%	–
BLSTM [9]	11.8%	3.7%	**11.9%**	**4.9%**
BLSTM [42]	12.3%	**3.3%**	13.6%	5.1%
Attribute SVMs [2]	–	–	20.01%	11.27%
Ours (Euclidean dist. + BoC)	20.82%	10.99%	20.91%	13.04%
Ours (Matching model + BoC)	10.87%	5.55%	**11.93%**	7.78%

word 'the' is much closer to the word 'they' than to 'thousand'. Similarly, the word 'three' is closer to 'there' than to 'they' or 'the'.

Table 7 shows the mean Average Precision for the 5, 10, 15, 20 and 25 first selected word images using the BoC-based model on the IAM testing set. The small differences between the mAP@5 and mAP@25 value, for both Euclidean distance and MLP matching model, demonstrates the stability of the proposed approach.

4.4.3. Word recognition

Table 8 compares the word recognition performance of our BoC-based model with that of recent approaches for this problem. Performance is measured in terms of WER and CER on test examples of the RIMES and IAM databases, using for each of these databases the combined words of the training and test sets as lexicon. Results show our method to outperform the approach of [2], which also employs a joint embedding strategy, as well the HMM-based techniques described in [6] and [14]. We note that a direct comparison to [14] is not possible since this work used images of handwritten lines instead of words. Furthermore, our method compares favorably to recent methods based on a combination of CNNs and Gaussian HMMs in tandem mode [8] or on Bidirectional LSTMs [9,42]. Hence, our method is within 3% of the best reported result, for all databases and performance metrics. While these approaches are tailored for word recognition, our method can also be employed for word spotting.

The word error rates and number of incorrect IAM word recognitions, for different word lengths, are shown in Fig. 6. Once again, our BoC-based approach performs better with short words than long ones. In particular, an important increase in error rate is observed for words with 4 or 5 characters. Inspecting errors on single-character words reveals that most of them are caused by

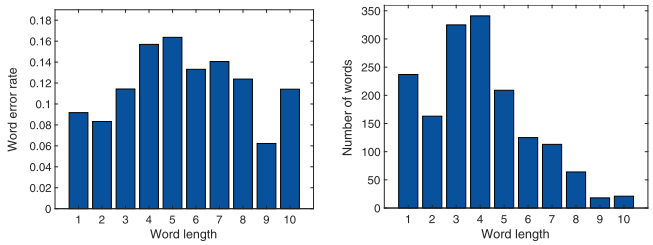


Fig. 6. The word error rates and number of incorrect word recognitions obtained by the BoC-based model, for different lengths of words in the IAM database.

similar-looking punctuation characters (i.e., '.' versus ',' or ':' versus ';').

4.4.4. Computational efficiency

For a given word image, the CNN branch of our architecture takes 1.8 ms on average to compute the embedding. Likewise, embedding word texts requires about 0.74 ms via the proposed RNN. For matching, computing the Euclidean distance between an embedded word and the embedding vector of all other words of the IAM test set (13K words) takes only 1.3 ms. In contrast, computing matching scores with the MLP model requires about 337 ms for the same word set.

In our experiments, we computed the Euclidean distance or matching score for all test examples in order to calculate the mAP. In large-scale word spotting or word recognition scenarios, a more efficient strategy could be used to find matching words. In this strategy, a short list of nearest-neighbors is first obtained using Euclidean distance. Let N be the number of word representations to compare against and M be the desired number of nearest-neighbors, this could be performed in $O(M \log N)$ operations via a specialized structure like kd-trees. Then, the MLP matching model would be applied only on this reduced list of candidates.

5. Conclusions

We presented a unified approach for solving the word spotting and recognition tasks. This approach embeds word images via a CNN preserving character-level information (e.g., BoC or PHOC encoding), and word texts using a bidirectional RNN which captures the spatial relationships between characters. A matching model based on multi-layer perceptrons was also proposed to evaluate the probability that two embedding vectors are from the same word.

Our method was tested on five public databases using the evaluation protocol of [2]. Results showed the advantage of employing a supervised matching model compared to simple Euclidean distance, for both word spotting and recognition. With respect to existing approaches, our method yielded state-of-the-art performance for query-by-example (QBE) word spotting, as well as very competitive performance for the query-by-string (QBS) scenario. The proposed method also provided good accuracy for word recognition, when compared against approaches tailored for this particular problem.

Although experiments indicate the learned representation to be transferable across datasets, this appears more challenging for databases containing dissimilar types of writing such as different languages. A potential extension of this work would be to explore domain adaptation techniques, for instance based on adversarial networks [52], to make the representation more robust to such differences. Moreover, our analysis revealed that most errors made by our method correspond to words of average length (i.e., 4 to 7 characters), possibly due their higher similarity. Investigating novel character-level encodings that are better suited for such words could thus improve performance.

Acknowledgments

The authors thank the NSERC (Grant no. RGPIN 2014-04649) of Canada for their financial support.

References

[1] J. Almazán, A. Gordo, A. Fornés, E. Valveny, Segmentation-free word spotting with exemplar SVMs, Pattern Recognit. 47 (12) (2014) 3967–3978.
[2] J. Almazán, A. Gordo, A. Fornés, E. Valveny, Word spotting and recognition with embedded attributes, IEEE Trans. Pattern Anal. Mach. Intell. 36 (2014) 2552–2566.
[3] Y. Bengio, Learning deep architectures for AI, Found. Trends Mach. Learn. 2 (2009) 1–127.
[4] Y. Bengio, A.C. Courville, P. Vincent, Representation learning: a review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell. 35 (2013) 1798–1828.
[5] A.K. Bhunia, P.P. Roy, A. Mohta, U. Pal, Cross-language framework for word recognition and spotting of indic scripts, Pattern Recognit. 79 (2018) 12–31.
[6] A.L. Bianne-Bernard, F. Menasri, R.A.-H. Mohamad, C. Mokbel, C. Kermorvant, L. Likforman-Sulem, Dynamic and contextual information in HMM modeling for handwritten word recognition, IEEE Trans. Pattern Anal. Mach. Intell. 33 (2011) 2066–2080.
[7] A. Bissacco, M. Cummins, Y. Netzer, H. Neven, PhotoOCR: reading text in uncontrolled conditions, in: IEEE Int. Conf. Comput. Vis., 2013, pp. 785–792.
[8] T. Bluche, H. Ney, C. Kermorvant, Tandem HMM with convolutional neural network for handwritten word recognition, in: IEEE Int. Conf. Acoust. Speech Signal Process., 2013, pp. 2390–2394.
[9] T. Bluche, H. Ney, C. Kermorvant, A comparison of sequence-trained deep neural networks and recurrent neural networks optical modeling for handwriting recognition, in: Int. Conf. Stat. Lang. Speech Process., 2014, pp. 199–210.
[10] Y.-N. Chen, D. Hakkani-Tür, X. He, Zero-shot learning of intent embeddings for expansion by convolutional deep structured semantic models, in: Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on, IEEE, 2016, pp. 6045–6049.
[11] J. Chung, C. Gülçehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, CoRRabs/1412.3555.
[12] J. Dolz, C. Desrosiers, I.B. Ayed, 3D Fully convolutional networks for subcortical segmentation in MRI: a large-scale study, Neuroimage (2017). abs/1612.03925
[13] R. El-Hajj, C. Mokbel, L. Likforman-Sulem, Recognition of arabic handwritten words using contextual character models, Document Recognit. Retrieval 6815 (2008) 681503.
[14] S. Espana-Boquera, M.J. Castro-Bleda, J. Gorbe-Moya, F. Zamora-Martinez, Improving offline handwritten text recognition with hybrid HMM/ANN models, IEEE Trans. Pattern Anal. Mach. Intell. 33 (2011) 767–779.
[15] G. Fink, T. Plotz, On the use of context-dependent modeling units for HMM-based offline handwriting recognition, in: International Conference on Document Analysis and Recognition, 2, 2007, pp. 729–733.
[16] A.P. Giotis, G. Sfikas, B. Gatos, C. Nikou, A survey of document image word spotting techniques, Pattern Recognit. 68 (2017) 310–332.
[17] A. Graves, S. Fernández, F. Gomez, Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks, in: Int. Conf. Mach. Learn., 2006, pp. 369–376.
[18] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, J. Schmidhuber, A novel connectionist system for unconstrained handwriting recognition, IEEE Trans. Pattern Anal. Mach. Intell. 31 (2009) 855–868.
[19] E. Grosicki, M. Carre, J.M. Brodin, E. Geoffrois, Results of the RIMES evaluation campaign for handwritten mail processing, in: Int. Conf. Document Anal. Recognit., 2009, pp. 941–945.
[20] L. Gómez, D. Karatzas, Textproposals: a text-specific selective search algorithm for word spotting in the wild, Pattern Recognit. 70 (2017) 60–74.
[21] R. Hadsell, S. Chopra, Y. LeCun, Dimensionality reduction by learning an invariant mapping, in: Comput. Vis. Pattern Recognit., 2006 IEEE computer society conference on, 2, 2006, pp. 1735–1742.
[22] K. He, X. Zhang, S. Ren, J. Sun, Spatial pyramid pooling in deep convolutional networks for visual recognition, IEEE Trans. Pattern Anal. Mach. Intell. 37 (2015) 1904–1916.
[23] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, CoRR (2015). abs/1502.03167
[24] M. Jaderberg, K. Simonyan, A. Vedaldi, A. Zisserman, Reading text in the wild with convolutional neural networks, Int. J. Comput. Vis. 116 (1) (2016) 1–20.
[25] M. Khayyat, L. Lam, C.Y. Suen, Learning-based word spotting system for arabic handwritten documents, Pattern Recognit. 47 (3) (2014) 1021–1030.
[26] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, CoRR (2014). abs/1412.6980.
[27] T. Konidaris, A.L. Kesidis, B. Gatos, A segmentation-free word spotting method for historical printed documents, Pattern Anal. Appl. 19 (2016) 963–976.
[28] P. Krishnan, K. Dutta, C.V. Jawahar, Deep feature embedding for accurate recognition and retrieval of handwritten text, in: International. Conference on Frontier and Handwriting Recognition (ICFHR), 2016, pp. 289–294.
[29] P. Krishnan, C.V. Jawahar, Matching handwritten document images, ECCV 2016 (2016) 766–782.

[30] M.R. L. Gomez, D. Karatzas, LSDE: Levenshtein space deep embedding for query-by-string word spotting, in: 017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), 2018, pp. 499–504.
[31] V.I. Levenshtein, Binary codes capable of correcting deletions, insertions and reversals, Soviet Phys. Doklady 10 (2012) 707.
[32] V.M.N.E.H.A. M. Pechwitz, S. Snoussi Maddouri, IFN/ENIT-Database of handwritten arabic words, Colloque International Francophone sur l'Ecrit et le Document (2002) 129–136.
[33] R. Manmatha, C. Han, E.M. Riseman, Word spotting: a new approach to indexing handwriting, Comput. Vis. Pattern Recognit. (1996) 631–637.
[34] R. Manmatha, J.L. Rothfeder, A scale space approach for automatically segmenting words from historical handwritten documents, IEEE Trans. Pattern Anal. Mach. Intell. 27 (8) (2005) 1212–1225.
[35] U. Marti, H. Bunke, The IAM-database: an english sentence database for off–line handwriting recognition, Int. J. Doc. Anal. Recognit. 5 (2002) 39–46.
[36] M. Mhiri, S. Abuelwafa, C. Desrosiers, M. Cheriet, Hierarchical representation learning using spherical k-means for segmentation-free word spotting, Pattern Recognit. Lett. 101 (2018) 52–59.
[37] M. Mhiri, M. Cheriet, C. Desrosiers, Query-by-example word spotting using multiscale features and classification in the space of representation differences, in: 2017 IEEE International Conference on Image Processing (ICIP), 2017, pp. 1112–1116.
[38] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, Adv. Neural Inf. Process. Syst. (2013) 3111–3119.
[39] R.A.-H. Mohamad, L. Likforman-Sulem, C. Mokbel, Combining slanted-frame classifiers for improved HMM-based arabic handwriting recognition, IEEE Trans. Pattern Anal. Mach. Intell. 31 (2009) 1165–1177.
[40] T. Mondal, N. Ragot, J.Y. Ramel, U. Pal, Comparative study of conventional time series matching techniques for word spotting, Pattern Recognit. 73 (2018) 47–64.
[41] L. Neumann, J. Matas, Scene text localization and recognition with oriented stroke detection, in: IEEE International Conference on Computer Vision, 2013, pp. 97–104.
[42] V. Pham, C. Kermorvant, J. Louradour, Dropout improves recurrent neural networks for handwriting recognition, CoRR (2013) 285–290.
[43] A. Poznanski, L. Wolf, CNN-N-gram for handwriting word recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2305–2314.
[44] J.A. Rodriguez-Serrano, F. Perronnin, A model-based sequence similarity with application to handwritten word spotting, IEEE Trans. Pattern Anal. Mach. Intell. (2012).
[45] P.P. Roy, A.K. Bhunia, A. Das, P. Dey, U. Pal, HMM-Based indic handwritten word recognition using zone segmentation, Pattern Recognit. (2016). abs/1708.00227
[46] M. Rusinol, D. Aldavert, R. Toledo, Browsing heterogeneous document collections by a segmentation-free word spotting method, in: International Conference on Document Analysis and Recognition, 2011, pp. 63–67.
[47] M. Rusinol, D. Aldavert, R. Toledo, J. Llados, Efficient segmentation-free keyword spotting in historical document collections, Pattern Recognit. 48 (2015) 545–555.
[48] A. Sharma, Adapting off-the-shelf CNNs for word spotting & recognition, in: Document Analysis and Recognition (ICDAR), 2015 13th International Conference on, 2015, pp. 986–990.
[49] S. Sudholt, G.A. Fink, PHOCNet: A deep convolutional neural network for word spotting in handwritten documents, CoRR (2016) 277–282.
[50] S. Sudholt, Evaluating word string embeddings and loss functions for CNN-based word spotting, in: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), 01, 2017, pp. 493–498.
[51] S. Sudholt, G.A. Fink, Attribute CNNs for word spotting in handwritten documents, Int. J. Doc. Anal.Recognit. (IJDAR) (2018) 1433–2825.
[52] E. Tzeng, J. Hoffman, K. Saenko, T. Darrell, Adversarial discriminative domain adaptation, Comput. Vis. Pattern Recognit. (CVPR) 1 (2) (2017) 4.
[53] A. Vinciarelli, J. Luettin, A new normalization technique for cursive handwritten words, Pattern Recognit. Lett. 22 (2001) 1043–1050.
[54] L. Wang, Y. Li, Learning deep structure-preserving image-text embeddings, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 5005–5013.
[55] T. Wilkinson, J. Lindström, Neural ctrl-F: segmentation-free query-by-string word spotting in handwritten manuscript collections, in: 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 4443–4452.
[56] M. Wöllmer, B. Schuller, G. Rigoll, Keyword spotting exploiting long short-term memory, Speech Commun. 55 (2) (2013) 252–265.

Mohamed Mhiri obtained his M.Sc. in Computer Engineering in 2014. Since september 2014, he is a Ph.D. student in Synchromedia Laboratory at École de Technologie Supérieure. He is expecting to finish his Ph.D. in august 2018. His main research interests focus on machine learning and pattern recognition.

Christian Desrosiers obtained a Ph.D. in Computer Engineering from Polytechnique Montréal in 2008, and was a postdoctoral researcher at the University of Minnesota on the topic of machine learning. In 2009, he joined ÉTS Montréal as professor in the Dept. of Software and IT Engineering. His main research interests focus on machine learning, image processing, computer vision and medical imaging. Dr. Desrosiers is codirector of the Laboratoire d'imagerie, de vision et d'intelligence artificielle (LIVIA) and is a member of the REPARTI research network.

Mohamed Cheriet received the M.Sc. and Ph.D. degrees in computer science from the University of Pierre and Marie Curie (Paris VI), Paris, France, in 1985 and 1988, respectively. Since 1992, he has been a Professor with the Department of Automation Engineering, École de Technologie Supérieure (University of Quebec), Montreal, QC, Canada, where he was appointed a Full Professor in 1998. He is the Founder and has been the Director since 1998 of the Synchromedia Laboratory at École de Technologie Supérieure, which targets multimedia communication in telepresence applications. He is an expert in computational intelligence, pattern recognition, mathematical modeling for image processing, and cognitive and machine learning approaches and perception. In addition, he has authored or coauthored over 350 technical papers in the field. Dr. Cheriet is the founder and a former Chair of the IEEE Montreal Chapter of Computational Intelligent Systems.