TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

BLSTM	Bidirectional Long Short-Term Memory
BoC	Bag of Character
CCA	Correlation Canonical Analysis
CER	Character Error Rate
CNN	Convolutional Neural Network
CTC	Connectionist Temporal Classification
DWT	Dynamic Time Warping
GMM	Gaussian Mixture Model
GRU	Gated Recurrent Units
GW	George Washington dataset
HMM	Hidden Markov Model
HoG	Histogram of Oriented Gradients
LB	LordByron dataset
LSTM	Long Short-Term Memory
mAP	mean Average precision
MLP	Multi-Layer Perceptron
PBCS	Pyramid of Bidirectional Character Sequences
PCA	Principal Component Analysis
PHOC	Pyramidal Histogram of Character
QBE	Query-by-Example
QBS	Query-by-String
RANSAC	RANdom SAmple Consensus
ReLU	Rectified Linear Unit

RNN	Recurrent Neural Network
SC-HMM	Semi-Continuous Hidden Markov Model
SIFT	Scale-Invariant Feature Transform
SVM	Support Vector Machines
TF-IDF	Term Frequency–Inverse Document Frequency
VLAD	Vector of Locally Aggregated Descriptors
VQ	Vector Quantization
WER	Word Error Rate

INTRODUCTION

0.1 Motivation

Les documents historiques sont une source précieuse d'information pour comprendre la culture et le patrimoine des civilisations passées. Afin d'offrir un large accès à leur contenu, les documents historiques doivent être transcrits dans un format numérique permettant l'analyse. Sans l'aide de l'ordinateur, la transcription manuelle de ces documents prendrait plusieurs décennies. Les systèmes automatiques de reconnaissance d'écriture manuscrite sont donc nécessaires pour préserver et étudier cet héritage. Ces systèmes sont également essentiels dans la vie quotidienne où les documents papier sont omniprésents. Par exemple, les formulaires et le courrier envoyés aux entreprises doivent être analysés de manière automatique afin de réduire le coût et le temps de leur traitement.

Dans cette thèse, nous nous concentrons sur la tâche de reconnaissance d'écriture manuscrite qui est au coeur de diverses applications d'analyse de documents numérisés. La reconnaissance peut être **explicite**, auquel cas le but est de reconnaître le texte correspondant à une image donnée, ou **implicite**, le but étant alors de repérer toutes les occurrences d'une requête texte dans une collection de documents numérisés.

La reconnaissance et le repérage d'écriture manuscrite sont des problèmes importants en recherche, comme en témoignent les nombreux projets centrés sur ceux-ci. Par exemple, les projets (*ANR*, 2007-2010) et (*ANR Digidoc*, 2011-2014) ont été lancé dans le but d'analyser le patrimoine écrit français. Ces projets focalisent sur l'acquisition de documents numérisés afin d'améliorer la reconnaissance automatique d'écriture manuscrite. Le projet (*PACTE*, 2011-2014) traite le problème de reconnaissance d'écriture dans les documents historiques multilingues, et vise l'amélioration de la reconnaissance à l'aide de modèles statistiques de langues. De même, le projet (*MAURDOR*,2010-2013) se concentre sur la reconnaissance d'écriture manuscrite en trois langues, l'anglais, le français et l'arabe, dans une base de documents récents.

Cette thèse s'inscrit dans le cadre d'un nouveau projet de traitement de documents historiques qui s'intitule (*VLP : Visual Language Processing of Ancient Manuscripts : Converting Collections to Windows on the Past*, 2014-2019). L'objectif principal de ce projet est de définir un graphe de similarité à partir d'une collection de plus de 20 millions documents historiques numérisés. Ce graphe sera utilisé par des historiens et sociologues pour analyser les relations et les interactions à travers les différents manuscrits, lieux et époques.

0.2 Problématique

Malgré les vastes efforts de recherche consacrés à la reconnaissance et le repérage d'écriture manuscrite, ces problèmes demeurent en grande partie non résolus. La difficulté à résoudre ces problèmes est liée à deux principaux facteurs :

— **La variabilité des formes manuscrites** qui est due au fait que chaque personne a son propre style d'écriture et que ce style peut varier d'un document à l'autre. Cette variabilité dépend aussi de l'état du document qui peut contenir diffèrents types de dégradations et bruits de fond.

— **La complexité du processus de reconnaissance et de repérage** composé d'une séquence d'étapes interdépendantes dans laquelle une erreur à une étape peut se propager aux étapes suivantes et affecter la performance du processus entier.

0.2.1 La variabilité de formes manuscrites

- **La variabilité intra-classe** des formes manuscrites a de nombreuses origines. Deux individus peuvent écrire un même texte de manières différentes. Les gens écrivent en utilisant de différents styles cursifs avec différentes tailles et formes alternatives. De même, un individu peut écrire le même texte différemment d'une fois à l'autre. L'état psychologique de l'auteur et la situation d'écriture sont ainsi des sources de variabilité. La Figure 0.1 présente le mot 'vous' écrit par différents personnes. On observe une importante variabilité intra-

classe sous la forme de déformations dans les traits, inexactitude dans les jonctions, parties manquantes et chevauchements, etc.

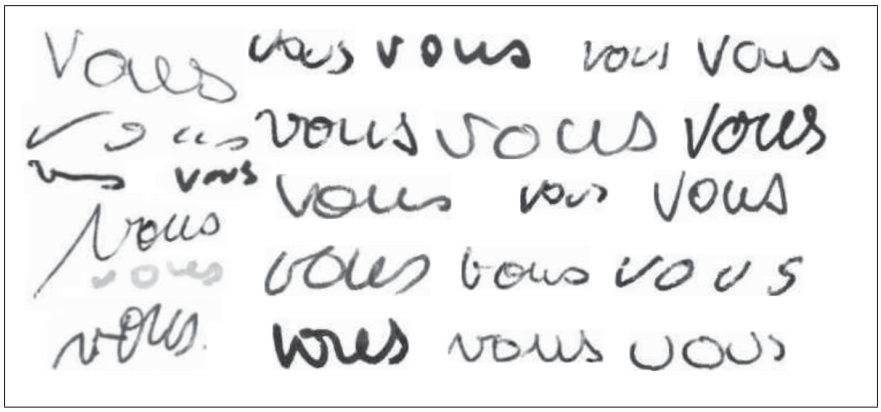


Figure 0.1 Le mot ‘vous’ écrit par différentes personnes (Bluche, 2015).

- **Le bruit dans les images** est souvent présent pour les documents historiques qui souffrent de différents types de dégradations en raison de l’âge et d’un mauvais entretien. Au cours du temps, la couleur du document change, l’encre diffuse dans le document et des taches peuvent apparaître. Dans le cas des formulaires et documents manuscrits plus récents, on peut retrouve d’autres formes de bruits comme les lignes de conduite imprimées qui servent à guider l’écriture dans le document. Différents types de bruits peuvent également survenir lors de la numérisation de documents, par exemple, la superposition du verso de pages telle que montrée à la Figure 0.2.

0.2.2 La complexité du processus de reconnaissance et de repérage

- **Une séquence d’étapes interdépendantes** doit être exécutée afin de passer d’un document numérisé à un texte pouvant être analysé. Tout d’abord, une analyse est effectuée sur les images de documents afin d’identifier les zones de texte. Par la suite, les zones de texte identifiées sont binarisées et segmentées en lignes et en mots. Le modèle de reconnaissance ou de repérage est ainsi appliqué sur les images de mots segmentés. Des ambiguïtés et

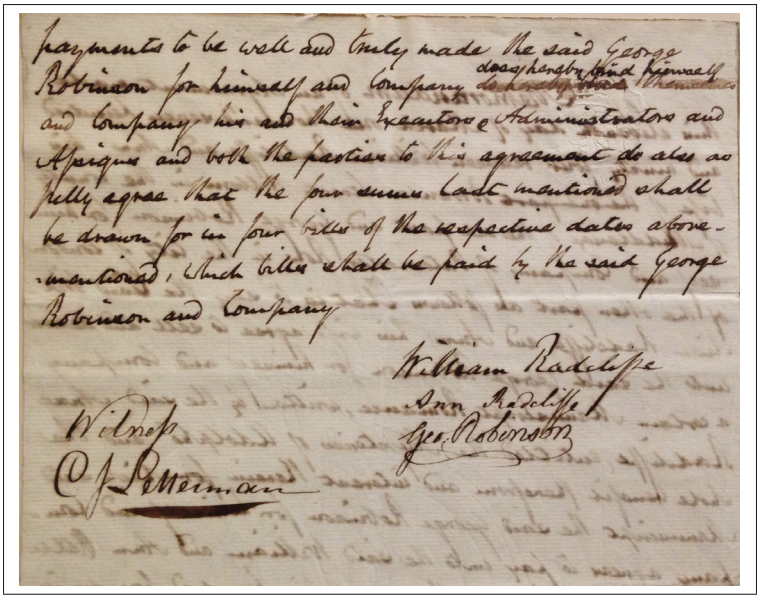


Figure 0.2 Un exemple d'une image d'un document avec de différents types de bruits.

des difficultés peuvent survenir à chaque étape du processus, pouvant affecter les étapes subséquentes.

- **La diversité des représentations pour les formes manuscrites** est un problème bien connu dans les modèles de reconnaissance et de repérage. De manière générale, ces représentations peuvent être classées selon leur niveau d'abstraction. Le plus bas niveau renferme des représentations utilisant la géométrie locale ou l'orientation du gradient local, tandis que les représentations haut niveau sont basées sur des caractéristiques symboliques, telles que la présence d'ascendants, de descendants, de points ou de boucles. Le choix d'utiliser une représentation ou une autre demeure jusqu'à maintenant une question ouverte puisque cette décision dépend aussi du modèle de reconnaissance ou de repérage utilisé.
- **Le manque de données étiquetées** est également un problème pour entraîner les modèles de reconnaissance et de repérage. L'obtention de telles données est coûteuse en raison du temps nécessaire pour effectuer l'étiquetage, surtout si les documents comportent plusieurs langues et différents styles d'écriture.

0.3 Objectif principal de la thèse

L'objectif principal de cette thèse est de concevoir un système efficace et robuste pour la reconnaissance et le repérage d'écriture manuscrite dans des documents numérisés. Dans le cadre de cette thèse, la recherche est concentrée sur le développement de modèles traitant des mots manuscrits isolés. Ce choix est motivé par le fait qu'un document peut être divisé en un ensemble de mots séparés à l'aide d'une méthode de segmentation. Le développement de telles méthodes de segmentation pourrait faire l'objet d'une autre recherche.

0.4 Contributions de la thèse

Afin d'atteindre cet objectif, il est nécessaire de définir des représentations robustes pour les formes manuscrites, pouvant surmonter les difficultés mentionnées à la section précédente. Une représentation robuste doit pouvoir distinguer entre les variabilités inter-classes et intra-classes de formes manuscrites. La séparation entre ces deux types de variabilité est une tâche très difficile qui ne peut pas être définie explicitement par un ensemble de règles. L'extraction intelligente et automatique de caractéristiques haut niveau est donc essentielle.

De nombreux travaux récents ont démontré les avantages des modèles d'apprentissage profond. Ces modèles permettent de capturer les caractéristiques haut niveau d'images via une hiérarchie de couches de traitement non linéaire. Dans cette hiérarchie, chaque couche est définie par un ensemble de paramètres reflétant un niveau plus élevé d'abstraction que ceux de la couche précédente. Des données étiquetées sont généralement utilisées pour apprendre ces paramètres lors d'une phase d'entraînement. Cependant, il existe également des modèles d'apprentissage profond non supervisés qui peuvent apprendre leurs paramètres sans avoir recours à des données étiquetées.

Dans ce contexte, les contributions majeures de cette thèse sont :

1. Une représentation profonde non supervisée (**Chapitre 2**) de formes manuscrites, permettant d'améliorer la tâche de repérage d'écriture. La représentation proposée se base sur

l'algorithme de regroupement 'spherical k-means', qui est employé pour définir une hiérarchie des fonctions paramétriques d'encodage. Les avantages de cette nouvelle représentation sont multiples. D'abord, elle est définie de manière automatique à partir d'images de documents non étiquetés, qui sont faciles à obtenir. De plus, cette représentation s'adapte mieux aux formes manuscrites ayant une importante variabilité inter-classe. Finalement, elle peut être calculée rapidement et offre une taille compacte,

Cette contribution de la thèse a mené à la publication de deux articles :

- **M. Mhiri**, S. Abuelwafa, C. Desrosiers, M. Cheriet, "Hierarchical representation learning using spherical k-means for segmentation-free word spotting", Pattern Recognition Letters, volume 101, 2018, pages 52-59.
- **M. Mhiri**, M. Cheriet, C. Desrosiers, "Query-by-example word spotting using multiscale features and classification in the space of representation differences", IEEE International Conference on Image Processing (ICIP), 2017, pages 1112-1116, .

2. Un modèle bout en bout pour la reconnaissance des mots manuscrits (**Chapitre 3**) pouvant faire, sans aucun pré-traitement, la correspondance entre l'image d'un mot et son texte. Ce modèle est composé d'un réseau de neurones convolutifs prenant en entrée une image et qui produit en sortie une représentation du texte reconnu.Ce texte est représenté sous la forme d'un ensemble de sous-séquences bidirectionnelles de caractères formant une hiérarchie. Cette représentation se distingue des approches existantes dans la littérature et offre plusieurs avantages par rapport à celles-ci. Notamment, elle est binaire et a une taille fixe, ce qui la rend robuste à la taille du texte. Par ailleurs, elle capture la distribution des sous-séquences de caractères dans le corpus d'entraînement, et permet donc au modèle entraîné de transférer cette connaissance à de nouveaux mots contenant les mêmes sous-séquences.

   Cette contribution a été publiée dans l'article suivant :

   - **M. Mhiri**, C. Desrosiers, M. Cheriet, "Convolutional pyramid of bidirectional character sequences for the recognition of handwritten words", Pattern Recognition Letters, volume 111, 2018, pages 87-93.

3. Un modèle bout en bout qui intègre conjointement les textes et les images de mots dans un seul espace vectoriel (**Chapitre 4**). Une image est projetée dans cet espace via un réseau de neurones convolutifs entraîné à détecter les différentes formes de caractères. De même, un mot est projeté dans cet espace via un réseau de neurones récurrents. Le modèle proposé est entraîné de manière à ce que l'image d'un mot et son texte soient projetés au même point. Dans l'espace vectoriel appris, les tâches de repérage et de reconnaissance peuvent alors être traitées efficacement comme un problème de recherche des plus proches voisins. Cette contribution a abouti à l'article suivant :

   - **M. Mhiri**, C. Desrosiers, M. Cheriet, "Word spotting and recognition via a joint deep embedding of image and text", Pattern Recognition, (accepté avec révision).

0.5 Plan de la thèse

La présente section a décrit le contexte, les objectifs et les contributions de cette recherche. La suite de la thèse est organisée comme suit.

- **Chapitre 1** fait une revue de la littérature sur les modèles de reconnaissance et de repérage d'écriture manuscrite, et discute les limitations de ces modèles.
- **Chapitre 2** décrit la représentation proposée pour le repérage de mots manuscrits. Ce chapitre correspond à l'article intitulé "Hierarchical representation learning using spherical k-means for segmentation-free word spotting", publié dans la revue Pattern Recognition Letters.
- **Chapitre 3** introduit ensuite le modèle de reconnaissance proposé pour faire la correspondance entre l'image d'un mot et son texte. Le contenu de ce chapitre correspond à l'article "Convolutional pyramid of bidirectional character sequences for the recognition of handwritten words", publié dans la revue Pattern Recognition Letters.
- **Chapitre 4** présente le modèle pour la reconnaissance et le repérage de mots intégrant les textes et les images de mots dans un même espace vectoriel. Ce chapitre correspond à l'article "Word spotting and recognition via a joint deep embedding of image and text", soumis au journal Pattern Recognition.

- Enfin, **Chapitre 5** fait la synthèse des contributions de la thèse, et discute les limitations et extensions possibles de cette recherche. Une liste complète des articles publiés dans le cadre de cette thèse y est également fournie.

CHAPITRE 1

REVUE DE LA LITTÉRATURE

1.1 Introduction

L'analyse automatique de l'écriture manuscrite est un problème recevant de plus en plus d'attention de la communauté de l'analyse de documents. Les deux principales tâches reliées à ce problème sont la reconnaissance et le repérage d'écriture manuscrite. Ces tâches sont souvent résolues en traitant des images de mots segmentés. La reconnaissance consiste alors à identifier le mot correspondant à une image donnée, et le repérage à trouver toutes les occurrences d'un mot requête dans une collection d'images de documents. Le mot requête peut être fourni sous la forme d'une image (problème de requête-par-image) ou un texte (problème de requête-par-texte).

La grande variabilité intra-classe de formes manuscrites et la faible qualité des documents numérisés sont les deux principaux facteurs qui rendent difficiles la reconnaissance et le repérage d'écriture manuscrite. La première section de ce chapitre présente les différentes étapes de pré-traitement appliquées aux documents numérisés. Ensuite, les sections 2 et 3 font respectivement une revue de littérature générale sur les modèles de reconnaissance et de repérage d'écriture manuscrite.

1.2 Le pré-traitement d'images de documents

Un document numérisé subit plusieurs étapes de pré-traitement pour arriver à un ensemble d'images de lignes et de mots. Ces images sont par la suite utilisées comme entrées pour les modèles de reconnaissance et de repérage d'écriture manuscrite.

1.2.1 La binarisation

Un problème crucial dans l'analyse de documents numérisés est la présence de différents types de dégradations. Généralement, les humains peuvent distinguer entre l'écriture manuscrite et les dégradations, cependant cette distinction demeure problématique pour les méthodes automatiques d'analyse. Au cours des dernières années, plusieurs méthodes de binarisation ont été proposées pour surmonter ce problème. Ces méthodes peuvent être catégorisées en deux catégories : les méthodes globales, qui emploient un seuil unique pour toute l'image, et les méthodes adaptatives (ou locales) qui considèrent un seuil différent pour chaque pixel de l'image.

La méthode d'Otsu est une méthode de binarisation avec un seuil global. Cette méthode suppose la présence de deux distributions dans une image d'un document, une pour l'avant-plan et l'autre pour l'arrière-plan. Le seuil global est calculé d'une manière à minimiser la variance intra-classe entre ces deux distributions. Cette méthode fonctionne bien avec les images éclairées, mais elle est moins intéressante pour les images dégradées où il y a une grande variance entre les pixels de l'avant-plan. La méthode de Niblack est une méthode avec seuil adaptatif qui permet de surpasser cette limitation. Dans cette méthode, le seuil pour un pixel p est calculé en utilisant une fenêtre centrée en p comme $S_{\text{Niblack}}(p) = m(p) + k\,\sigma(p)$, où $m(p)$ et $\sigma(p)$ sont la moyenne locale et l'écart-type dans la fenêtre centrée. Cependant, cette méthode peut considérer des pixels comme avant-plan si le seuil est trop petit, ce qui limite sa performance dans certains cas.

La méthode de Sauvola est une approche populaire de binarisation qui évite les limitations de Niblack en supposant que l'avant-plan et l'arrière-plan ont des intensités proches de 0 et 255. La méthode d'Howe (Howe, 2013) offre également une bonne performance en traitant la binarisation comme un problème d'optimisation d'une fonction d'énergie globale qui dépend de six paramètres dont deux sont définis de manière automatique.

1.2.2 La segmentation en zones, lignes et mots

Une image binarisée de document doit généralement être segmentée en un ensemble de régions classifiées en zones de texte et zones de non-texte. Les zones de texte sont par la suite segmentées en un ensemble de lignes. Cette étape est délicate (Gatos & Pratikakis, 2009) puisque les lignes ne sont pas toujours horizontales et les descendants/ascendants de différentes lignes peuvent se chevaucher. Une étude comparative des différentes méthodes pour la segmentation de lignes est faite dans (Likforman-Sulem *et al.*, 2007; Louloudis *et al.*, 2009).

Une fois extraites, les lignes doivent ensuite être segmentées en un ensemble des mots afin de pouvoir être traitées directement. La segmentation en mots passe se base généralement sur l'analyse de composants connectés ou des profils de projection verticale (Gatos & Pratikakis, 2009; Louloudis *et al.*, 2009) pour trouver les séparations entre les mots. Les différentes étapes nécessaires pour passer d'un document numérisé à un ensemble d'images de mots sont illustrées à la Figure 1.1.

1.2.3 Discussion

La correction de la pente d'écriture et de l'angle d'inclinaison de caractères sont deux tâches fréquemment employées pour normaliser l'écriture manuscrite. La binarisation peut également être considérée comme une opération de normalisation. Ces trois étapes de sont appliquées afin de réduire la variabilité de l'écriture manuscrite. Cependant, elles ne fonctionnent pas toujours parfaitement et leur utilisation peut causer la perte d'informations importantes. Dans cette thèse, nous proposons de modèles qui n'appliquent **aucune étape de normalisation**. Le but est d'exploiter plus efficacement l'apprentissage automatique de manière à éviter ces pertes.

1.3 Les modèles de reconnaissances de mots manuscrits

La reconnaissance de mots manuscrits est le processus qui consiste à identifier le mot écrit correspondant à une image donnée. Dans la littérature, cette tâche est largement abordée par les chercheurs. Les premiers travaux étaient limités à la reconnaissance d'images de caractères et

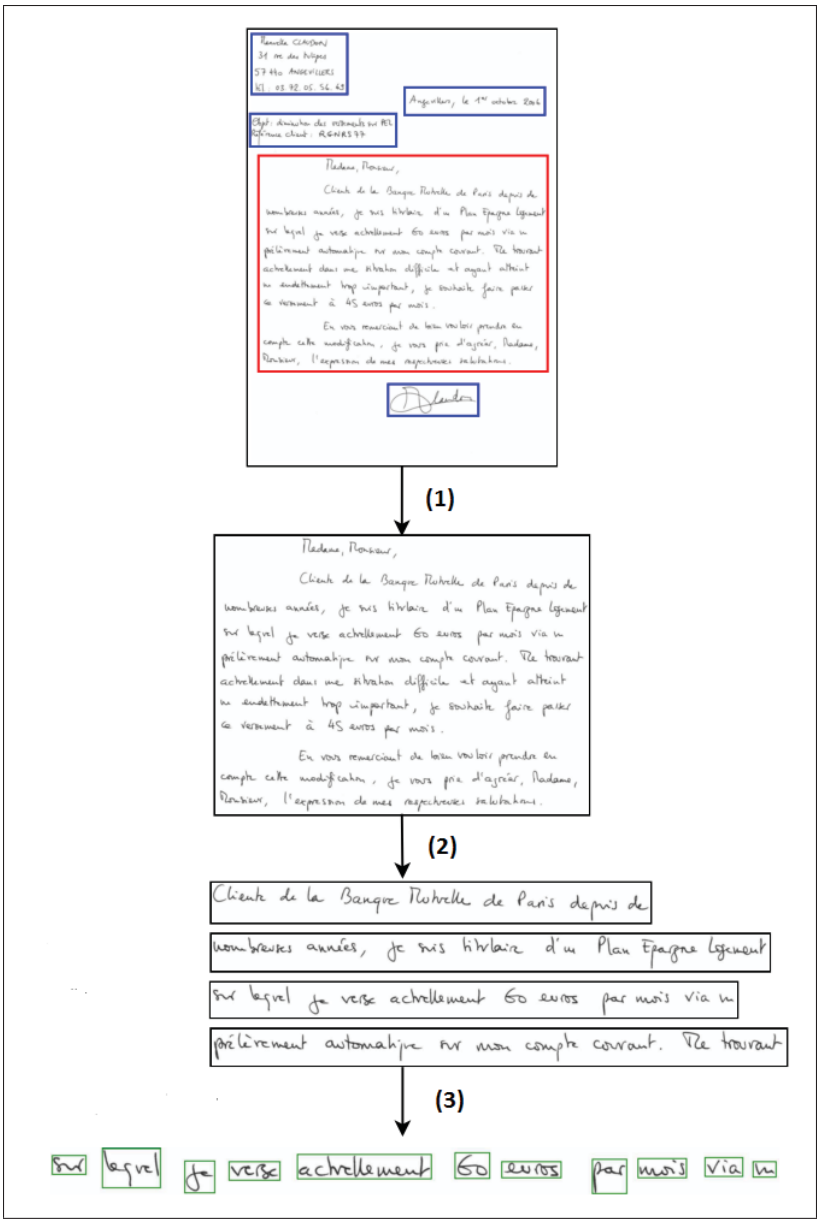


Figure 1.1 Les différentes étapes appliquées pour passer d'une image d'un document à un ensemble d'images de mots. 1) Segmentation en zones ; 2) Segmentation en lignes ; 3) Segmentation en mots.

de chiffres (Lecun *et al.*, 1998). Par la suite, les modèles de reconnaissance se sont améliorés et la recherche s'est concentrée sur aux problèmes plus complexe de reconnaître des mots entiers ou des lignes. Les approches pour ces problèmes sont généralement basées sur des modèles de données séquentielles tels que les modèles de Markov cachés et les modèles récurrents.

Récemment, les réseaux de neurones convolutifs sont également devenus populaires pour la reconnaissance d'écriture manuscrite.

1.3.1 Les modèles de Markov cachés

Un modèle de Markov caché (*Hidden Markov Model* ou HMM) décrit un processus stochastique composé d'une séquence d'états cachés qui sont observés indirectement à travers un autre processus. L'état du modèle à un instant t ne dépend que de l'état à l'instant $t-1$. À chaque instant, des observations sont émises selon une fonction de densité de probabilité dépendant de l'état courant. Formellement, un modèle de Markov caché de K états $\{S_1,\ldots,S_K\}$ est défini par trois mesures de probabilités $\lambda = (\pi, A, B)$ comme suit.

- π : les probabilités initiales, où π_i est la probabilité que S_i soit l'état initial.
- A : la matrice de transition d'états, où a_{ij} est la probabilité de transition de l'état S_i à l'état S_j.
- B : la probabilité d'émission, où $b_i(o)$ est la probabilité d'émettre l'observation o étant dans l'état S_i.

Une fois les paramètres $\lambda = (\pi, A, B)$ appris, le modèle peut par la suite être utilisé pour calculer la probabilité $P(O \mid \lambda)$ d'une séquence donnée d'observations $O = \{o_1,\ldots,o_T\}$. Pour la reconnaissance d'écriture manuscrite, un modèle de Markov caché est défini pour chaque caractère de l'alphabet. Ces modèles sont ensuite combinés pour produire des modèles de mots. Étant donné l'image d'un mot représentée sous la forme d'une séquence d'observations (Figure 1.2), le mot reconnu correspond au modèle donnant la plus grande probabilité pour la séquence.

La reconnaissance à l'aide de modèles de Markov cachés a l'avantage d'être hors-vocabulaire, c'est-à-dire qu'un mot peut être reconnu sans avoir besoin d'être segmenté en caractères au préalable. Toutefois, l'inconvénient majeur de cette approche est que la modélisation de caractères est faite sans tenir compte de leur contexte dans les mots (Vinciarelli & Luettin, 2001; Mohamad *et al.*, 2009). Dans la littérature, plusieurs travaux (Fink & Plotz, 2007; El-Hajj *et al.*,

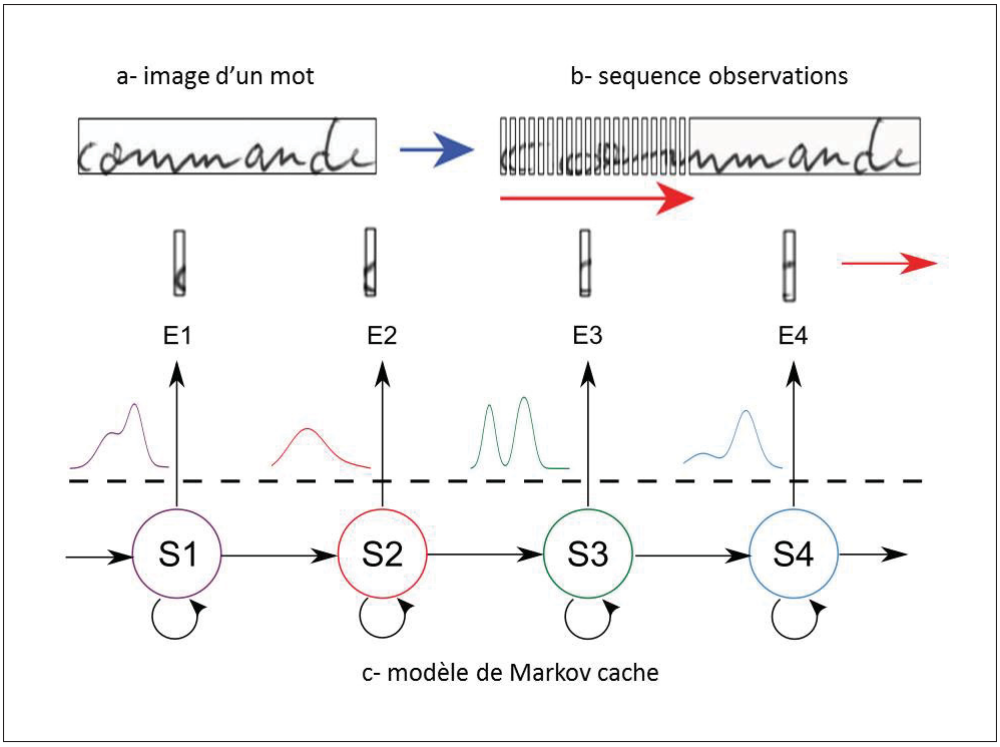


Figure 1.2 La reconnaissance avec les modèles de Markov cachés. a) L'image d'un mot ; b) La séquence d'observations ; c) Un modèle de Markov caché utilisé pour calculer la probabilité d'un mot par rapport à la séquence d'observations.

2008) se sont penchés sur cette limitation. Une solution est de considérer le contexte d'un caractère comme les caractères l'entourant. Cette technique conduit cependant à un nombre élevé de modèles, chacun avec de nombreux paramètres à apprendre, et ne convient pas lorsque les données d'entraînement sont limitées.

1.3.2 Les modèles convolutifs

Les modèles convolutifs (*Convolutional Neural Networks* ou CNN) ont montré des performances exceptionnelles pour diverses tâches de traitement d'images. Contrairement aux approches séquentielles, les modèles convolutifs peuvent représenter et manipuler les images de manière plus efficace (Bengio, 2009; Bengio *et al.*, 2013). Les modèles de reconnaissance convolutifs peuvent être regroupés en deux grandes catégories :

- **Les modèles de reconnaissance caractère-par-caractère :** (Alsharif & Pineau, 2013; Jaderberg *et al.*, 2014b) les caractères sont reconnus séparément et une étape d'inférence est appliquée par la suite pour trouver le mot le plus probable.
- **Les modèles de reconnaissance de mots entiers :** (Goodfellow *et al.*, 2013; Jaderberg *et al.*, 2016, 2014a) la reconnaissance consiste à trouver le mot correspondant à une image d'entrée sans reconnaître explicitement ses caractères. Généralement, une représentation est définie pour chaque mot du dictionnaire et le modèle convolutif apprend la correspondance entre les images et leur représentation.

Dans (Alsharif & Pineau, 2013), un modèle de reconnaissance caractère-par-caractère est présenté. Dans ce modèle, l'image est initialement sur-segmentée en régions de caractères candidats. Le mot le plus probable est ensuite trouvé en combinant une technique de reconnaissance de caractères basée sur un réseau de neurones convolutifs et un modèle de Markov caché pour trouver le mot le plus probable. De même, Jaderberg et al. (Jaderberg *et al.*, 2014b) ont proposé un modèle de reconnaissance caractère-par-caractère qui combine un classificateur binaire pour détecter les régions de caractères, un réseau de neurones convolutifs pour reconnaître ces régions, et l'algorithme de Viterbi pour trouver la séquence la plus probable.

Récemment, quelques travaux (Goodfellow *et al.*, 2013; Jaderberg *et al.*, 2016, 2014a) ont considéré les modèles convolutifs pour la reconnaissance entière d'une image de mot. Dans (Goodfellow *et al.*, 2013), un réseau de neurones convolutifs sensible à la position est utilisé pour reconnaître les nombres à plusieurs chiffres dans les images de rue en supposant que les nombres ont une longueur maximale fixe. Jaderberg et al. (Jaderberg *et al.*, 2016) ont défini la reconnaissance comme une tâche de classification dans un ensemble de mots d'un dictionnaire. Dans (Jaderberg *et al.*, 2014a), un réseau de neurones convolutifs qui contient plusieurs classificateurs indépendants est proposé. Chaque classificateur prédit le caractère à une position spécifique dans le mot. Un mot est ensuite reconnu en trouvant le mot du lexique le plus proche de la séquence de caractères prédite.

1.3.3 Les modèles récurrents

Le principal inconvénient majeur d'utiliser les modèles de Markov cachés pour la reconnaissance d'écriture manuscrite est leur limitation à modéliser la dépendance à long terme dans une séquence de données. Les modèles récurrents, qui ne souffrent pas de cette limitation, sont des réseaux de neurones adaptés pour traiter les séquences de données grâce à des connexions récurrentes. Les réseaux récurrents avec mémoires à court et long termes (*Long short-term memory* ou LSTM) sont une approche populaire pour la reconnaissance d'écriture. Grâce à leur architecture, ces modèles sont particulièrement efficaces pour préserver l'information contextuelle dans les séquences.

Pour entraîner un réseau LSTM à étiqueter une séquence d'entrée sans segmentation explicite de ses caractères, la méthode de classification temporelle connexionniste (CTC, (Graves *et al.*, 2006)) est souvent utilisée. Cette méthode repose sur une procédure avant-arrière (*forward-backward*) pour transformer une séquence S en une séquence d'étiquettes L qui représente le texte reconnu (Figure 1.3). En somme, un réseau LSTM produit une séquence S contenant différentes étiquettes parmi les caractères d'un alphabet, en plus du caractère spécial 'vide' ($\emptyset$). La méthode CTC transforme par la suite la séquence S à la séquence L obtenue en supprimant d'abord les répétitions d'étiquettes, puis les étiquettes 'vides'. Par exemple, la séquence $\{\mathtt{a}, \mathtt{a}, \emptyset, \emptyset, \mathtt{b}, \mathtt{b}, \emptyset, \mathtt{b}, \mathtt{a}\}$ qui est la sortie du réseau LSTM sera transformée en la séquence $\{\mathtt{a}, \mathtt{b}, \mathtt{b}, \mathtt{a}\}$ par la méthode CTC.

1.3.4 Discussion

Les modèles de Markov cachés, récurrents et convolutifs ont montré de bons résultats pour la tâche de reconnaissance d'écriture manuscrite. Chacun de ces modèles possède cependant des forces et faiblesses. Ainsi, les modèles convolutifs peuvent extraire de manière intelligente des caractéristiques haut niveau à partir d'images. En revanche, les modèles de Markov cachés et récurrents sont mieux adaptés pour représenter et traiter les séquences.

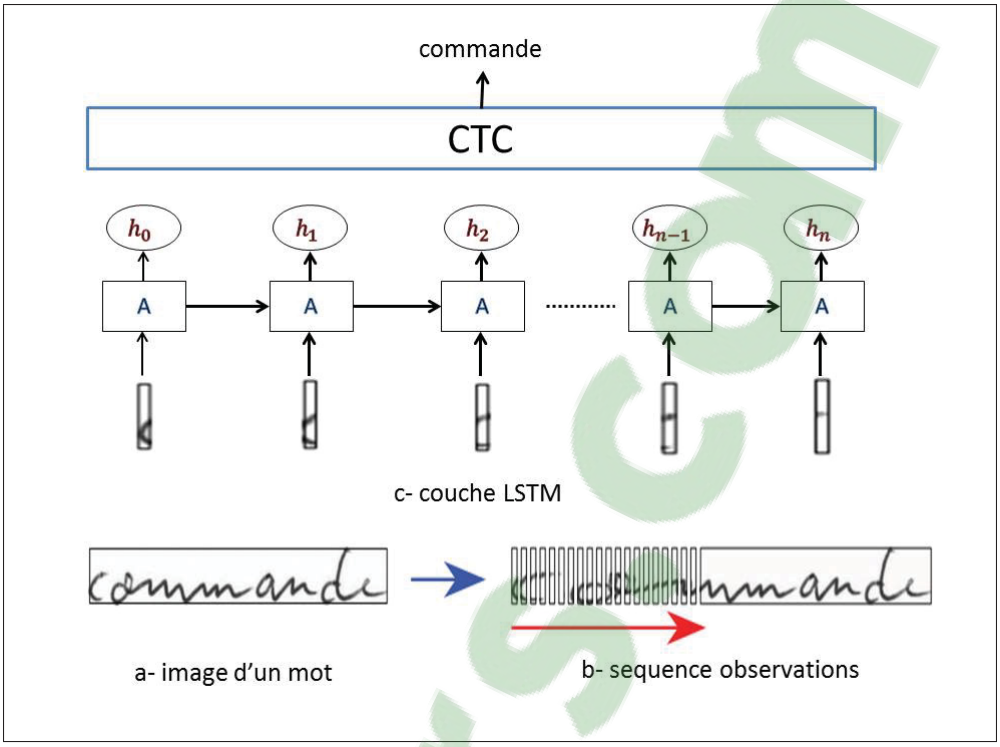


Figure 1.3 La reconnaissance avec les modèles récurrents. a) L'image d'un mot ; b) La séquence d'observations ; c) Un modèle récurrent composé d'une couche LSTM suivi par une couche CTC qui produit le texte reconnu.

Bluche et al. (Bluche *et al.*, 2013a) ont démontré que la combinaison d'un modèle convolutifs pour l'apprentissage de caractéristiques avec un modèle de Markov caché pour la reconnaissance de mots surpassait le modèle de Markov caché basé sur des caractéristiques traditionnelles. De même, Shi et al. (Baoguang Shi, 2015) ont proposé un modèle avec des performances remarquables qui combine des couches convolutives, plusieurs couches LSTM et une couche de transcription CTC. Cette thèse propose des **modèles à la fois convolutifs et récurrents** afin de profiter des avantages complémentaires de ces deux approches.

1.4 Les modèles de repérage de mots manuscrits

La reconnaissance d'écriture manuscrite est souvent employée pour l'analyse de documents numérisés. Par contre, cette tâche peut être coûteuse en termes de calculs. De plus, l'entraîne-

ment du modèle nécessite une grande quantité de données étiquetées qui ne sont pas toujours disponibles (Manmatha *et al.*, 1996; Fischer, 2012). Comme alternative, une recherche par requête peut être utilisée (Manmatha *et al.*, 1996; Rodriguez-Serrano & Perronnin, 2012). Le but de cette recherche est de repérer toutes les occurrences d'un mot-requête dans une collection de documents numérisés. Le mot-requête peut être une image (requête-par-image) ou un texte (requête-par-texte).

1.4.1 Les différentes catégories de modèles de repérage

De manière générale, les méthodes de repérage peuvent être regroupées selon plusieurs critères (Rodriguez-Serrano & Perronnin, 2012; Fischer *et al.*, 2012; Rath & Manmatha, 2007; Pratikakis *et al.*, 2016).

- **Supervisé ou non supervisé** : les modèles de repérage supervisés sont entraînés avec de données étiquetées. Ils sont plus coûteux que les modèles non supervisés, car l'obtention de données étiquetées est une tâche fastidieuse. En revanche, les modèles non supervisés sont plus pratiques, mais leurs performances sont encore insuffisantes pour des applications réelles.
- **Avec segmentation ou sans segmentation** : les modèles de repérage sans segmentation sont appliqués directement sur des images de documents tandis que les modèles avec segmentation nécessitent que les mots soient préalablement segmentés. Généralement, les modèles de repérage avec segmentation sont moins pratiques puisque la segmentation de mots est une tâche difficile. À l'inverse, les modèles sans segmentation sont plus lents étant donné que l'image entière d'un document doit être traitée.

1.4.2 Les modèles de repérage avec requête-par-image

Le repérage avec requête-par-image se base principalement sur la similarité visuelle entre les images. Au cours des années, plusieurs représentations d'images ont été proposées pour cette tâche (Rusinol *et al.*, 2011; Almázan *et al.*, 2014b; P. Keaton & Goodman, 1997; Rusinol *et al.*,

2015; Gatos & Pratikakis, 2009; Konidaris *et al.*, 2016). Ces représentations doivent pouvoir capturer la grande variabilité intra-classe des formes manuscrites, être robustes aux bruits et dégradations présents dans les documents, et pouvoir être calculées et comparées rapidement.

Les représentations d'images de documents (Rusinol *et al.*, 2011; Almázan *et al.*, 2014b; P. Keaton & Goodman, 1997; Rusinol *et al.*, 2015; Gatos & Pratikakis, 2009; Konidaris *et al.*, 2016) peuvent employer sur des caractéristiques traditionnelles ou des caractéristiques apprises (Graves *et al.*, 2009a; Slimane *et al.*, 2012). Les caractéristiques traditionnelles sont largement utilisées car elles sont faciles à interpréter. Cependant, les caractéristiques apprises permettent de mieux représenter les données et sont généralement plus performantes (Bengio, 2009; Bengio *et al.*, 2013).

Keaton et al. (P. Keaton & Goodman, 1997) représentent l'image d'un mot par un ensemble de caractéristiques basées sur des projections horizontales et verticales. Rusinol et al. (Rusinol *et al.*, 2011) divisent l'image en une grille de sous-régions, chacune représentée par un sac-de-mots-visuels (Csurka *et al.*, 2004) défini à l'aide d'un ensemble de descripteurs SIFT. Almazan et al. (Almázan *et al.*, 2014b) utilisent une représentation de sous-régions basée sur les histogrammes de gradient orienté (HoG). Sudholt et al. (Sudholt & Fink, 2016) encodent l'image d'un mot à l'aide d'un modèle convolutif. Cette représentation, appelée *Pyramidal Histogram of Characters* (PHOC) (Almázan *et al.*, 2014a), représente les occurrences de N-grammes (uni-grammes, bi-grammes et tri-grammes) dans le mot et sous-parties de ce mot.

1.4.3 Les modèles de repérage avec requête-par-texte

Le repérage avec requête-par-texte est souvent préféré au repérage avec requête-par-image. Dans ce cas, l'utilisateur fournit seulement une chaîne de caractères alors qu'il a besoin de chercher une image requête lors du repérage avec requête-par-image. Cependant, l'inconvénient du repérage avec requête-par-texte est le besoin d'avoir de données étiquetées pour entraîner un modèle faisant la correspondance entre la requête texte et les images de mots.

Les modèles de données séquentielles, tels que les modèles de Markov cachés et les modèles récurrents, sont souvent utilisés pour le repérage avec requête-par-texte. L'inconvénient principal de ces modèles est le temps de calcul élevé pour comparer la requête texte avec les images de mots. Récemment, Almazan et al. (Almázan *et al.*, 2014a) ont défini un espace de représentation commun pour les images et les textes (les chaînes de caractères). Le repérage est par la suite traité comme une recherche du plus proche voisin dans cet espace appris. Dans cette méthode, les images de mots sont représentées par des vecteurs de Fisher et les chaînes de caractères par des vecteurs PHOC. Par la suite, une méthode d'analyse de corrélation canonique (CCA) est utilisée pour intégrer ces vecteurs dans un même espace vectoriel.

1.4.4 Discussion

Concevoir une représentation efficace pour les formes manuscrites est une tâche essentielle à la robustesse des modèles de repérage. Ainsi, une représentation efficace doit pouvoir modéliser la variabilité intra-classe des formes manuscrites, doit être robuste aux bruits et dégradations, tout en étant rapide à calculer et comparer. Sans une étape d'apprentissage, une telle représentation peut être difficile à définir.

D'un autre coté, définir un espace de représentation commun pour les images de mots et les chaînes de caractères (Almázan *et al.*, 2014a) est une idée très prometteuse. Dans cet espace, les deux tâches de repérage avec requête-par-image et avec requête-par-texte peuvent être résolues simultanément. Dans cette thèse, les modèles d'apprentissage profond seront investigués afin d'apprendre un tel espace. L'apprentissage peut se faire à partir de données étiquetées ou d'une façon plus indépendante, à l'aide de données non étiquetées.

CHAPITRE 2

HIERARCHICAL REPRESENTATION LEARNING USING SPHERICAL K-MEANS FOR SEGMENTATION-FREE WORD SPOTTING

Mohamed Mhiri[1], Sherif Abuelwafa[1], Christian Desrosiers[1], Mohamed Cheriet[1]

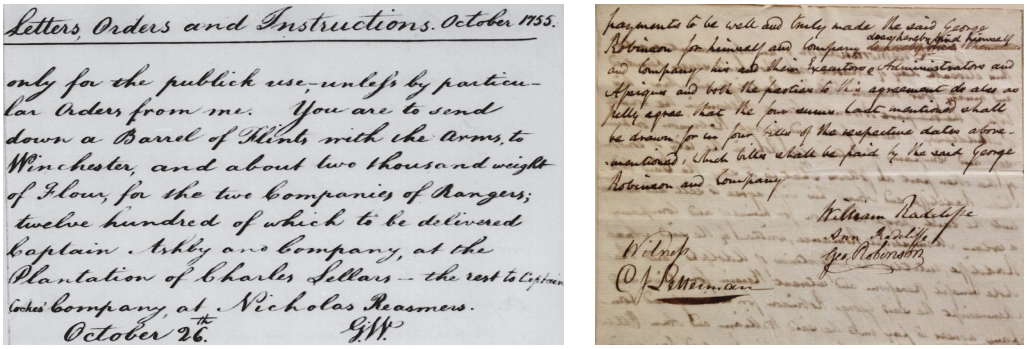
[1] Département de génie de la production automatisée, École de Technologie Supérieure, 1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3

Article publié à la revue « Pattern recognition letters », volume 101, pages 52-59, 2018.

2.1 Introduction

The number of digitized historical documents is increasing every day. In order to analyze these documents and facilitate their use, handwriting recognition is often applied as pre-processing step. Unfortunately, this process can be expensive, requiring both time and labeled data, and can be insufficient to understand the handwritten documents (Manmatha *et al.*, 1996; Fischer, 2012). As an alternative, query by example image-based search can be used (Manmatha *et al.*, 1996; Rodriguez-Serrano & Perronnin, 2012). Given a word image, the main objective of query by example word spotting is to find all occurrences of this image in a dataset of digitized documents. However, handwriting variability makes this task quite challenging, in particular for multi-writer datasets. Additionally, the existence of various types of degradations makes this task even more prone to errors. For instance, Figure 2.1 presents two examples of ancient document images illustrating the problems of handwriting variability and degradation. Finally, the lack of labeled data is another obstacle when training the matching system (Youssouf, 2016).

Numerous word spotting methods have been proposed in the literature. In general, these methods can be grouped depending on whether they need training data or require words to be segmented (Rodriguez-Serrano & Perronnin, 2012; Fischer *et al.*, 2012; Rath & Manmatha, 2007; Pratikakis *et al.*, 2016) :



(a) (b)

Figure 2.1 Examples showing the main challenges of word spotting in ancient document images : (a) Document image with significant handwriting variability (i.e., character variability). (b) Document image in degraded condition.

- **Training-based vs training-free :** Training-based (or *supervised learning*) systems need labeled data to learn a model during training. Such systems are typically more expensive than training-free methods since obtaining labeled data can be a hard and time-consuming task. In contrast, training-free (or *unsupervised learning*) systems are more convenient but their performances are still insufficient for real applications.
- **Segmentation-based vs segmentation-free :** While segmentation-free systems can be applied directly on full document images, segmentation-based approaches require individual words to be previously segmented from these full images. Segmentation-based systems are less practical considering that the problem of word segmentation is as challenging as word spotting. On the other hand, segmentation-free systems are often slower since the whole document image must be processed.

In this paper, we tackle the query by example word spotting task in a segmentation-free and training-free multi-writers scenario. In this scenario, given a set of document images and a target word image, the goal is to find all regions in the document images where this word occurs. The contributions of this work are twofold. First, we propose an unsupervised hierarchical handwriting representation, where the spherical k-means algorithm is used to learn a hierarchy of features. Secondly, we present an efficient matching system composed of a fast pre-selection

stage and a discriminative re-ranking stage. In a comprehensive experimental validation involving three different datasets, we show our method to yield competitive performance compared to state-of-the-art approaches for segmentation-free and training-free word spotting.

The rest of this paper is organized as follows. In Section 2.2, we present recent work on word spotting and explain the different techniques used to represent handwriting data. In Section 2.3, we then detail our proposed approach. Afterwards, in Section 3.3, our method is validated on three different datasets and compared to state-of-the-art approaches for this task. Finally, we conclude with a brief discussion of this work's contributions.

2.2 Related work

To process handwriting images efficiently, these images should be encoded in a representation that can model the large intra-class variability of handwritten shapes and differentiate between the target classes (Chherawala *et al.*, 2013). An effective representation must also adapt to any language and handwriting style, while being robust to the noise and degradation that may exist in the images. Lastly, it must be able to perform computations and comparisons efficiently. Defining a handwriting representation that satisfies all these properties is a challenging task.

Over the years, various representations have been proposed in the literature (Rusinol *et al.*, 2011; Almázan *et al.*, 2014b; P. Keaton & Goodman, 1997; Rusinol *et al.*, 2015; Gatos & Pratikakis, 2009; Konidaris *et al.*, 2016). Most of them concentrate on the critical step of feature representation, which can be divided in two categories : handcrafted features and learned features (Graves *et al.*, 2009a; Slimane *et al.*, 2012). Handcrafted features are widely used since they are easy to implement and interpret. However, recent works have shown the advantages of learned features (Bengio, 2009; Bengio *et al.*, 2013), which capture properties of the data in a more automated way and can better adapt to the target task than handcrafted features.

In the next sub-section, we present some recent handwriting representation techniques used for the task of segmentation-free and training-free word spotting. We then describe related works on unsupervised representation learning based on clustering.

2.2.1 Segmentation-free and training-free word spotting methods

Several handwriting representations have been proposed for the tasks of segmentation-free and training-free word spotting in document images. In an early work by Keaton et al. (P. Keaton & Goodman, 1997), handwriting is represented by a set of features based on horizontal and vertical projections of a word image, and by local features in the form of profile signatures. To match similar words, the Dynamic Time Warping (DWT) algorithm is used (P. Keaton & Goodman, 1997). Since the DWT algorithm is computationally expensive, this method does not scale well to large datasets. In contrast, the work of Rusinol et al. (Rusinol *et al.*, 2011) focused on the task of word spotting in large datasets. In this work, each document image is divided into a grid of equally-sized overlapping patches. Then, each patch is described using the bag-of-visual-words model (Csurka *et al.*, 2004) over extracted SIFT descriptors. This representation is refined with a latent semantic indexing technique (Deerwester *et al.*, 1990), and the cosine distance is used to rank the selected regions.

Similarly, the work of Almazan et al. (Almázan *et al.*, 2014b) used a histogram-based representation. In this work, the word image is divided into even-sized patches, each one represented by Histogram of Oriented Grandients (HOG) features. Histograms are then compressed with Product Quantization (Jegou *et al.*, 2011) to save memory and allow a sliding window-based search to be applied efficiently over a large dataset of document images. During the matching process, Almazan et al. used an exemplar SVM (Malisiewicz *et al.*, 2011) to decide whether a selected window is similar to the target word image or not.

Recently, Rusinol et al. proposed an improved version of their approach in (Rusinol *et al.*, 2011), achieving large performance improvements (Rusinol *et al.*, 2015). While their previous approach extracts SIFT descriptors in three different scales to capture fine and coarse features, the improved method uses a multi-sized patch representation (i.e., instead of a one-sized patch in (Rusinol *et al.*, 2011)), providing more robustness to the handwriting representation. Each patch is then described with a spatial bag-of-visual-words model using the extracted SIFT

descriptors. After this, a post-processing is applied, where the patch descriptors are re-weighted and normalized using the TF-IDF model (Salton & Buckley, 1988) and the L_2 norm.

The work of Gatos et al. in (Gatos & Pratikakis, 2009) also exploits a sliding-window approach, in which a patch-based representation is used to encode pixel densities. To find similar word images, an expensive template matching process is applied to a limited number of regions of interest. In a related work, Konidaris et al. (Konidaris *et al.*, 2016) perform word spotting in a two-step process. First, a minimum distance matching is used to select candidate regions and reduce the search space. Then, each selected candidate region is matched with the query word image representation (i.e., match the SIFT keypoints). The RANSAC algorithm (Fischler & Bolles, 1981) is employed to select the final bounding boxes.

In this study, we compare our word spotting method with the approaches proposed by Almazan et al. (Almázan *et al.*, 2014b) and Rusinol et al. (Rusinol *et al.*, 2011, 2015). These approaches were used for comparison since they are considered state-of-the-art for segmentation-free and training-free word-spotting, and because the datasets on which they were evaluated are publicly available.

2.2.2 Hierarchical representation learning using spherical k-means

Because the number of image pixels can be large, extracting high-level features from pixels is necessary to have an efficient handwriting representation. Hierarchical models like deep neural networks can learn such features in a data-driven manner, via sequential layers of non-linear processing (Bengio, 2009; Bengio *et al.*, 2013). Generally, these models can be divided in two categories : supervised models, which use labeled data to learn high-level features, and unsupervised models, which are more human independent and can learn high-level features from unlabeled data (Coates *et al.*, 2011b; Coates & Ng, 2011).

Unsupervised feature learning models can tackle the training-free word spotting task when only unlabeled examples are available. This is done by defining a feature representation θ that parametrizes an encoding function $\Phi_\theta(\mathbf{x})$. By applying Φ to the patches covering the

whole document image, a high-level representation can thus be defined. A more sophisticated description can also be built by composing multiple encoding functions in successive layers, i.e. $\left(\Phi^{L}_{\theta_L} \circ \cdots \circ \Phi^{1}_{\theta_1}\right)$, where L is the number of layers and $\{\theta_1, \ldots, \theta_L\}$ are the learned features (i.e., parameters) of these layers. Commonly, a spatial pooling operation is added between each two consecutive layers to progressively reduce the spatial size of the representation (i.e., number of parameters) and make it invariant to small translations (Bengio, 2009; Bengio *et al.*, 2013).

In (Csurka *et al.*, 2004; Sivic & Zisserman, 2003; Lazebnik *et al.*, 2006), the number of feature components was shown to be a crucial parameter in defining robust representations. Likewise, Van Gemert et al. (van Gemert *et al.*, 2008) have found that a soft encoding function tends to work better than a hard assignment. Moreover, a study by Boureau et al. (Boureau *et al.*, 2010; lan Boureau *et al.*, 2010) compared several architectures with different encoding functions (hard, soft and sparse coding) and pooling functions (average and maximum), showing max pooling to yield remarkable performance. Lastly, the importance of using non-linear encoding functions with a local contrast normalization has been demonstrated in the work of Jarret et al. (Jarrett *et al.*, 2009), where different types of normalization and rectification were studied.

A flurry of unsupervised feature learning algorithms have been proposed in the literature to learn the representation θ, including techniques based on sparse-coding (Scholkopf *et al.*, 2007), RBMs (Hinton *et al.*, 2006), sparse RBMs (Lee *et al.*, 2008) and denoising auto-encoders (Vincent *et al.*, 2008). Recently, a variant of the k-means algorithm called *spherical k-means* was shown to be an efficient alternative to these techniques (Coates & Ng, 2011). This algorithm provides a highly over-complete set of features (Coates *et al.*, 2011b), which can describe effectively the input data space. In addition, the implementation of the spherical k-means algorithm is simple, making this approach computationally efficient. Unlike k-means, the spherical k-means algorithm is based on cosine similarity, which performs better than the Euclidean distance for high-dimensional data (Aggarwal *et al.*, 2001).

2.3 Methodology

The proposed word spotting method has two main components. The first component, illustrated in Figure 2.2, is an unsupervised hierarchical representation based on the spherical k-means algorithm, for encoding document images into discriminative features. The second component, shown in Figure 2.3, uses this encoding in a multi-step word spotting process. Given the feature representation of a document image, a compression technique is first applied to speed-up computations. Afterwards, a sliding window is used through the compressed representation to detect regions similar to the query word's representation (unlike patches which have a fixed size, regions have the same size as the query word). Candidate regions are then ranked according to their approximated Euclidean distance in the representation space. Finally, to improve the efficiency of the proposed approach, we perform a re-ranking step applying a second ranking on the M first retrieved regions in order of relevance (Wu *et al.*, 2011). For this step, we use a discriminative representation based on the VLAD method (Jegou *et al.*, 2010), which can encode the spatial layout information of the local features. The following sections give a detailed description of these two components.

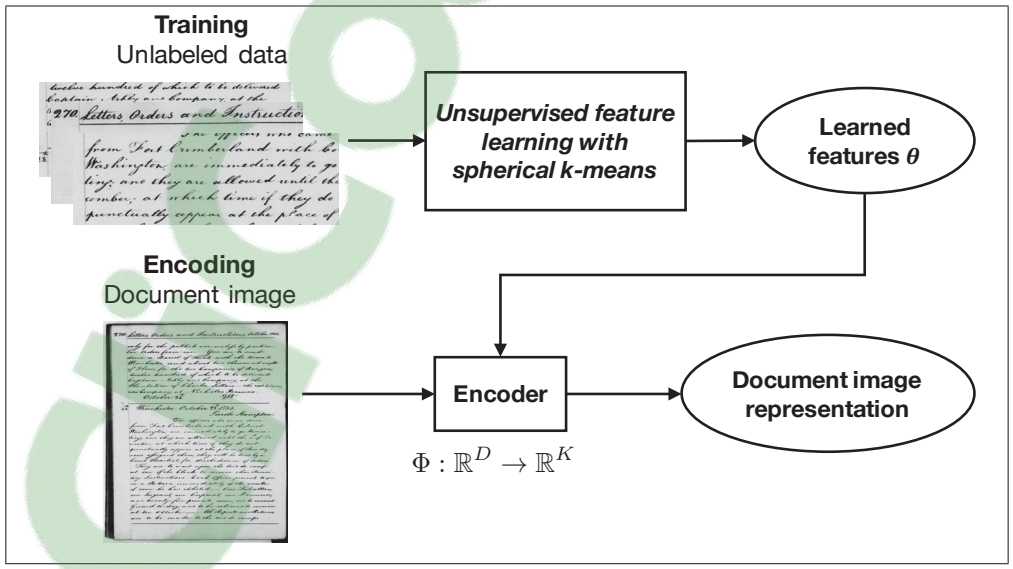


Figure 2.2 An unsupervised representation system used to represent the document images composed of one layer.

2.3.1 Unsupervised hierarchical features using spherical k-means

The unsupervised hierarchical representation employed in our method is composed of two stages : feature representation learning and feature encoding. First, a set of patches is extracted randomly from the input data and preprocessed. Then, the preprocessed patches are used as input to the spherical k-means algorithm (Hornik *et al.*, 2012) to learn the feature representation θ. In a later stage, these learned features are used to define an encoding function Φ, which maps an input image to the representation space. The following two sub-sections provide additional details about these two stages.

2.3.1.1 Feature representation learning

Learning the feature representation θ, used by the encoding function Φ in a later stage, requires a set of patches to be extracted randomly from a set of unlabeled input data. The extracted patches are then vectorized and preprocessed using principal component analysis (PCA) (Bengio, 2009; Bengio *et al.*, 2013) to remove the linear correlations between the nearby patch descriptors. We denote by the $\mathbf{X}$ the matrix of preprocessed patches, each column corresponding to a patch of size $D = N \times N \times$ channels, where channels $= 1$ for gray-level images. Following this, the spherical k-means algorithm is applied to learn the parameters θ, corresponding to the cluster centroids obtained by partitioning the set of patches into K clusters. For presentation purposes, we suppose θ to be the matrix whose columns are the cluster centroids.

The spherical k-means algorithm performs the following steps to learn the feature representation θ :

1. For each $\mathbf{x}_i$, compute assignment weight z_{ik} for each cluster θ_k (i.e., a single element will be non-zero) :

$$z_{ik} := \begin{cases} \theta_k^\top \mathbf{x}_i, & \text{if } k = argmax_j |\theta_j^\top \mathbf{x}_i| \\ 0, & \text{otherwise,} \end{cases} \tag{2.1}$$

2. Recompute θ by projecting $\mathbf{Z}$ on $\mathbf{X}$:

$$\theta \ := \ \mathbf{XZ}; \tag{2.2}$$

3. Unit normalize the columns of θ :

$$\widetilde{\theta}_k \ := \ \theta_k / \|\theta_k\|_2, \ \ k = 1, \ldots, K; \tag{2.3}$$

These steps are repeated until convergence is reached. As the k-means algorithm, this algorithm is guaranteed to converge, although not necessarily to a global optimum (Coates *et al.*, 2011b; Coates & Ng, 2011).

2.3.1.2 Feature encoding

After running spherical k-means to learn parameters θ from the unlabeled patches, a mapping function is needed to map a patch $\mathbf{x}_i$ to a corresponding feature vector $\mathbf{y}_i$. In this work, we used the soft-threshold function (Coates *et al.*, 2011b; Coates & Ng, 2011; Krizhevsky & Hinton, 2009), which is commonly employed in many feature learning architectures due to its computational efficiency and its role in regularization. The soft threshold function is defined by

$$\mathbf{y}_i \ = \ \max\left(0, \theta^\top \mathbf{x}_i - \alpha\right), \tag{2.4}$$

where $\alpha \geq 0$ is a parameter controlling the level of feature sparsity.

The encoding process can be summarized in the following steps :

1. Define a feature encoding function Φ by learning the parameter θ and by fixing the threshold α (in this work, α is fixed to 0) ;
2. Extract features using Φ from patches covering the given input data entirely (i.e., the input data is divided into $N \times N$ patches, separated by a step size of S) ;

3. After encoding, pool the resulting features over fixed regions to reduce the representation size and make this representation invariant to small translations.

The next section explains how this encoding can be used efficiently through a process of compression and re-ranking.

2.3.2 Compression and re-ranking using a codebook of local features

In the scenario of segmentation-free word spotting, a sliding-window based approach is commonly used. In the presence of large representations, running a sliding window throughout the document image can significantly slow down the word-spotting process. An efficient solution to this problem is to compress the representation, this technique also enabling a large collection of compressed document images to be stored.

Another commonly-used technique in segmentation-free word spotting on a large scale dataset is to apply a second ranking step on the M first retrieved regions (Pedronette *et al.*, 2014; Wu *et al.*, 2011). For this propose, a codebook is learned to compress the document images representation and, simultaneously, define a more discriminative description for re-ranking.

First, the local features $\mathbf{y}_i$ obtained by the encoding function are used with the k-means algorithm to learn a set of cluster centroids $\mathscr{C} = \{\mathbf{c}_j\}$, $j = 1, \ldots, C$, called *codebook*. The obtained codebook is then used with the product quantization (PQ) technique (Jegou *et al.*, 2011) to compress features $\mathbf{y}_i$. Furthermore, for re-ranking, this codebook is applied with the VLAD representation (Jegou *et al.*, 2010) to define a vector of small dimension that can preserve the spatial layout information of features. Spatial information is encoded by concatenating VLAD vectors computed in different regions of the image. The following sub-sections provide more details on these two steps.

2.3.2.1 Compression of document image representations

To compress a given image document representation, each local feature $\mathbf{y}_i$ (i.e., each patch features) is quantized to its nearest centroid in terms of the Euclidean distance, using the following equation :

$$q(\mathbf{y}_i) \;=\; argmin_{\mathbf{c}_j \in \mathscr{C}} \; \|\mathbf{y}_i - \mathbf{c}_j\|_2, \tag{2.5}$$

where $\mathbf{c}_j$ is the j-th codebook vector. The distance between region $\mathbf{W}^1 = [\mathbf{y}_1^1, \ldots, \mathbf{y}_P^1]$ and region $\mathbf{W}^2 = [\mathbf{y}_1^2, \ldots, \mathbf{y}_P^2]$, both composed of P patch features, is then approximated by comparing the vectors based on their quantization index :

$$\widehat{d}(\mathbf{W}^1, \mathbf{W}^2) \;=\; \sqrt{\sum_{j=1}^{P} d\Big(q(\mathbf{y}_j^1), q(\mathbf{y}_j^2)\Big)^2}. \tag{2.6}$$

To speed-up the computation, distances between centroids are pre-calculated.

2.3.2.2 Re-ranking using a sequence of VLAD representations

For re-ranking, we represent the query word image and the M first selected regions using a more discriminative description. Encoding the position of local features is extremely important to describe word images, since it provides knowledge about the location of characters in the word. On the other hand, the aggregation of local features is suitable to reduce the variability. Therefore, our proposed representation is an aggregation of local features, which can retain implicitly the spatial layout information inside the word image. It is based on the VLAD representation (Vector of Locally Aggregated Descriptors), which is considered as a simplification of the Fisher kernel representation (Jegou *et al.*, 2010).

In VLAD, for a given word image representation composed of P local features $\mathbf{W} = [\mathbf{y}_1, \ldots, \mathbf{y}_P]$, each local patch representation is assigned to the nearest visual word. The VLAD representation is a vector $\mathbf{v}$ where each element is the sum of differences between a visual word $\mathbf{c}_j$ and the local features assigned to this word. Let $\mathscr{C}_j$ be the indexes of local features assigned to the

j-th visual word, the VLAD vector $\mathbf{v}$ can be defined element-wise as

$$v_j = \sum_{i \in \mathscr{C}_j} \|\mathbf{c}_j - \mathbf{y}_i\|_2 \tag{2.7}$$

Finally, the vector $\mathbf{v}$ obtained from Eq. (2.7) is unit normalized.

In the proposed re-ranking representation, the word image representation is first divided in $W \times H$ regions. Then, each region is encoded using a VLAD vector. Lastly, the final vector representation is the concatenation of the $W \times H$ obtained VLAD vectors, as shown in Figure 2.3.

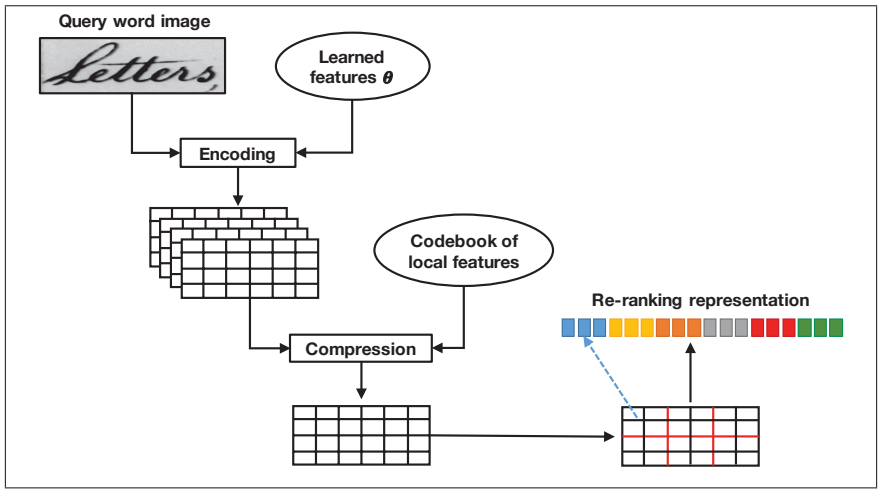


Figure 2.3 The proposed re-ranking representation, which is the concatenation of $w \times h$ (e.g., w=3 and h=2) independent VLAD vectors.

2.4 Experiments

2.4.1 Experimental setup

To evaluate our approach in training-free and segmentation-free scenario, two datasets are first used : the George Washington dataset (Rusinol *et al.*, 2011), which contains 20 pages of historical documents and includes 4860 word instances written by two persons, and the Lord

Byron (LB) dataset (Rusinol *et al.*, 2011), which contains 20 pages of typewritten text and includes 4988 word instances. In both datasets, each word image is considered as a query. From every document image, the 1000 most similar regions to the query image are then selected and ranked. If two selected regions are overlapping (overlap more than 10%), then only the most similar region is kept. Finally, the selected regions from all the document images are grouped together and ranked according to their distance to the query word image. A selected region is considered as positive (relevant) if it overlaps more than 50% with a bounding-box in the documents images containing the same query word. Performance is evaluated in terms of mean Average Precision (mAP), which is a standard measure in retrieval systems and is defined as follows :

$$\text{mAP} = \frac{1}{Q}\sum_{q=1}^{Q}\text{AP}_q \tag{2.8}$$

$$\text{AP}_q = \frac{1}{R_q}\sum_{k=1}^{N_q}P(k)\times rel(k) \tag{2.9}$$

where

– Q is the number of the queries ;

– AP_q is the average precision for the q^{th} query word image ;

– N_q is the number of the selected regions that are similar to the q^{th} query word image ;

– $rel(k)$ is 1 if the rank k is a relevant region and 0 otherwise ;

– $P(k)$ is the precision at cut-off k in the list ;

– R_q the number of relevant regions for the q^{th} query word image.

2.4.2 Effect of handwriting representation parameters

The proposed handwriting representation is a parameterized technique. The number of layers, the number of features in each layer, the patch size and the step size between two patches are various adjustable parameters. In this section, we analyze the impact of these parameters on word spotting performance. To perform such analysis, various configurations are tested. For a

given configuration, the representation of all document images is computed. To facilitate and to speed up the experimentations, these representations are then compressed using a codebook of size C=64. The impact of this compression process is evaluated in the next section.

Tableau 2.1 Performance in terms mAP and runtime obtained for different configurations of a single layer. The time is only reported for the GW dataset since both datasets are similar. Note : 2×2 max pooling is used for all tested configurations.

Patch size (N)	6	8	8	8	12	12	12
Step size (S)	**3**	**4**	4	4	**4**	**6**	6
Number of features (K)	128	128	**256**	**512**	128	128	512
mAP (GW)	**52.2%**	49.4%	49.9%	50.1%	48.3%	37%	38.6%
mAP (LB)	**85.9%**	76.1%	77.4%	78.3%	75.7%	61.6%	63.1%
Time query/doc (GW)	61ms	26ms	26ms	26ms	26ms	5ms	5ms

Table 2.1 gives the performance in terms of mAP obtained for different configurations of a single layer. To compare the execution time, all experiments were run using the same hardware setting. Results shows that increasing the step size between patches (i.e., parameter S) decreases significantly the word spotting performance : a loss of approximately 10% in the LB dataset and 3% in the GB dataset when going from S=3 to S=4, and more than 10% in both datasets when going from S=4 to S=6. In state-of-the-art methods, a step size of S=1 or S=2 is commonly used (Coates *et al.*, 2011b; Coates & Ng, 2011). However, using these small step size values with a single layer makes the document image representation sizable, which affects the time to process the whole representation.

The number of features also affects the word spotting performance. Using too few features leads to a weak representation that cannot discriminate between the different examples. In contrast, having a larger number of features, obtained through non-linear projections of the data, provides a more discriminative representation. Similarly, increasing the patch size generally decreases performance, the best results obtained for N=6. In summary, a good single-layer configuration has a small step size and a representation involving a large number of features.

The number of layers is an important parameter that controls the depth of the hierarchical representation. In Table 2.2, we give the performance obtained for two different configurations, each having two layers. We observe that using two layers of learned features performs better than a single layer. The first two-layer configuration in this table can be compared to the first single-layer configuration since these two configurations produce representations with similar sizes. We see that the two-layer configuration outperforms the single-layer one by approximately 5% in the GW dataset and 3% in the LB dataset. The same observation can be made when comparing the second two-layers configuration with the second single-layer configuration.

Tableau 2.2 Performance in terms mAP and runtime obtained for different configurations with two layers. Note : 2×2 max pooling is used for all tested configurations.

First layer **Second layer**	N**=8,** S**=4,** K**=512** $\varnothing$	N**=4,** S**=2,** K**=64** N**=3,** S**=1,** K**=512**	N**=12,** S**=6,** K**=512** $\varnothing$	N**=6,** S**=3,** K**=128** N**=3,** S**=1,** K**=512**
mAP (GW)	50.1%	**55.7%**	38.6%	**42.8%**
mAP (LB)	78.3%	**81.2%**	63.1%	**65.4%**
Time query/doc (GW)	26ms	26ms	5ms	5ms

2.4.3 Effect of compression

In previous experiments, vector quantization (VQ) (Jegou *et al.*, 2011) is used to compress the document image representations, leading to a faster execution of the sliding-window approach. In this section, we evaluate the effect of compression on the accuracy and runtime of the proposed method. The runtime (in milliseconds) is the average time needed to scan a single document image representation using a sliding window approach. Table 2.3 presents the results for increasing codebook sizes C.

Tableau 2.3 Performances in terms mAP and runtime obtained for a single-layer configuration with N=8, S=4, K=512 and 2×2 max pooling, using different codebook sizes C.

Codebook size	C**=32**	C**=64**	C**=128**	**Without VQ**
mAP (GW)	**49.4%**	50.1%	50.6%	**51.3%**
mAP (LB)	**79.7%**	78.3%	78.9%	**79.3%**
Time query/doc (GW)	26ms	26ms	26ms	310ms

When using VQ, the word spotting performance suffers a small mAP drop of 2%. On the other hand, applying VQ makes the sliding-window approach 12 times faster, thus allowing more time for the re-ranking step. Moreover, the memory needed to encode the document image representations becomes much lower with VQ since each $N \times N$ patch of pixels is encoded simply by the index of its nearest centroid.

2.4.4 Effect of re-ranking

In the compression process, each local feature vector (i.e., patch features) is quantized to its nearest centroid. For re-ranking, the distance from these feature vectors to their nearest centroid is used to define a more discriminative representation, as explained in Section 2.3.2.2. This re-ranking process is controlled by four parameters : the size of the codebook C, the number M of regions to re-rank, and the grid parameters (W,H). In Table 2.4, we set C=64, H=2 and M=20, and measured the impact of re-ranking on word spotting performance for various values of W.

Tableau 2.4 Performance in terms of mAP obtained with or without re-ranking (using different values of W when applying re-ranking) for two different configurations. Single-layer configuration : N=8, S=4, K=512, 2×2 max pooling. Two-layer configuration : (layer 1) N=4, S=2, K=64, 2×2 max pooling, (layer 2) N=3, S=1, K=512, 2×2 max pooling.

Configuration		**Without re-ranking**	**With re-ranking**			
			W=2	**W=5**	**W=9**	**W=12**
Single-layer	(GW)	**50.1%**	44.7%	48.2%	**51.7%**	50.9%
	(LB)	**78.3%**	73.5%	75.3%	77.6%	**78.1%**
Two-layer	(GW)	**55.7%**	46.7%	53.2%	**57.9%**	56.3%
	(LB)	**81.2%**	73.5%	75.3%	80.8%	**81.1%**

From Table 2.4, we see that the best performance is obtained for W=9. Using W=2 or W=5, the spatial layout information (i.e., the information of the local features inside the word image representation) may be poorly represented. For W=12, aggregation occurs in smaller sub-regions that better capture spatial layout information but affects the resilience to the shape variability. The improvement in performance with re-ranking is about 2% in the GW dataset, whereas a

loss of 1% is observed for the LB dataset. This difference in performance can be explained by the nature of the datasets, the LB dataset having less shape variability.

2.4.5 Comparison to the state-of-the-art

In Table 2.5, our proposed method is compared to different segmentation-free word spotting approaches (Rusinol *et al.*, 2011, 2015; Almázan *et al.*, 2014b) on the GW and LB datasets. Once again, performance is calculated in terms of mean Average Precision (mAP). We observe that the proposed method outperforms the approach of Rusinol et al. (Rusinol *et al.*, 2011), for both the GW and LB datasets. Similarly, our method yields better performance than the approach of Almazan et al. (Almázan *et al.*, 2014b) based on Exemplar SVM. This improvement could be due to the sequence of pooling functions in our method, which alleviates the impact of handwriting shape variability.

Tableau 2.5 Performance in terms of mAP obtained by the proposed method compared to state-of-the-art word spotting approaches. A two-layers configuration is used for our method : (layer 1) N=4, S=2, K=64, 2×2 max pooling, (layer 2) N=3, S=1, K=512, 2×2 max pooling.

Method	**GW**	**LB**
Rusinol et al. (Rusinol *et al.*, 2011)	30.42%	42.83%
Almazan et al. (Cosine distance + noPQ) (Almázan *et al.*, 2014b)	48.66%	74.04%
Almazan et al. (Exemplar SVM + PQ) (Almázan *et al.*, 2014b)	51.88%	84.34%
Almazan et al. (Exemplar SVM + PQ + query expansion + re-ranking) (Almázan *et al.*, 2014b)	59.13%	84.04%
Rusinol et al. (Rusinol *et al.*, 2015)	61.35%	**90.38%**
Proposed (Euclidean dist. + PQ)	55.7%	81.2%
Proposed (Euclidean dist. + PQ + re-ranking)	57.9%	81.1%
Proposed (Binarization + Euclidean dist. + PQ + re-ranking)	**62.3%**	87.1%

By introducing a binarization step based on Sauvola's method (Sauvola *et al.*, 1997), a better performance is achieved by our method. The proposed representation uses patches extracted from raw document images, which are sensitive to degradations and differences in pixel intensities. Applying binarization can thus reduce this sensitivity. In their work, Rusinol et al. (Rusinol *et al.*, 2015) proposed a representation based on SIFT features, which offers robustness to degradations without the need for preprocessing. Our method outperforms this approach

in the GW dataset but not in the LB dataset. This could be explained by the fact that the GW dataset has more intra-class variability than the LB dataset, hence benefits more from our method's pooling operations.

Memory analysis. In the proposed representation (i.e., the two-layer configuration with N=4, S=2, K =64, 2×2 max pooling for layer 1, and N=3, S=1, K=512, 2×2 max pooling for layer 2), two consecutive 2×2 max-pooling operations are applied with a step size of S=2 in the first layer and S=1 in the second layer. This leads to a 8×8 pixels representation after quantization by a one byte integer (i.e., the index of the nearest centroid). In the work of Almazan el al. (Almázan *et al.*, 2014b), HoG descriptors are extracted form patches of 12×12 pixels. Each HoG descriptor is then quantized using 6 local sub-vectors, giving 6 bytes for each 12×12 pixels (i.e., an average of 2.66 bytes for each 8×8 pixels). Moreover, in the approach proposed by Rusinol et al. (Rusinol *et al.*, 2015), features are extracted from patches of 27×27 pixels and each local feature vector is quantized using 128 sub-vectors, which leads to an average of 11.23 bytes per 8×8 pixels. Thus, our method offers a higher memory efficiency than these state-of-the-art approaches (Almázan *et al.*, 2014b; Rusinol *et al.*, 2015).

Execution time analysis. All our experiments were conducted on a single core 3.40 GHz Intel-i7 processor, using C++ code generated from MATLAB scripts. Using the two layer configuration of Table 2.5, the average time to process a single document image representation for a given query is about 26 ms. This is faster than the work of Almazan et al. (Almázan *et al.*, 2014b), which reported a time of 86 millisecond using a 2.67 GH core, but slower than the work of Rusinol et al. (Rusinol *et al.*, 2015) where a time of 3.01 ms was reported.

Multi-writers scenario. We used the multi-writer IAM dataset (Marti & Bunke, 2002) to evaluate the proposed approach in a more challenging setting. This dataset contains 115,320 labeled words from 657 different writers, where only a subset of 96,456 labeled word images are correctly segmented. Following the evaluation protocol of (Almázan *et al.*, 2014a), the pre-defined testing set, which 13,753 word images corresponding to 2,904 distinct words, was

employed to measure performance. Only queries with at least two occurrences were considered.

After binarizing the word images with Sauvola's method (Sauvola *et al.*, 1997), the following two-layer configuration was considered : N=4, S=2, K=64, 2×2 max-pooling for the first layer, and N=3, S=1, K=512, with 2×8 max-pooling for the second layer. Each local feature vector (i.e., each region features) was then quantized to its nearest centroid in terms of the Euclidean distance, using the vector quantization (VQ) technique (Jegou *et al.*, 2011) with a codebook of size C=256.

Tableau 2.6 Performance in terms of mAP obtained on the IAM dataset with the following two-layer configuration : (layer 1) N=4, S=2, K=64, 2×2 max pooling, and (layer 2) N=3, S=1, K=512, 2×8 max pooling.

Method	**Type**	**mAP**
Fisher Vector representation (Almázan *et al.*, 2014a)	Training-free	15.66%
DTW + Vinciarelli features (Almázan *et al.*, 2014a)	Training-free	12.30%
Proposed representation (spherical k-means)	Training-free	**28.31%**
Attribute based representation (Almázan *et al.*, 2014a)	**Training-based**	**55.73%**

As shown in Table 2.6, the proposed representation using spherical k-means outperforms the Fisher Vector representation based on the statistics of Gaussian mixtures learned from SIFT descriptors. Similarly, our method outperforms the dynamic time warping (DTW) approach based on Vinciarelli features (Vinciarelli & Bengio, 2002).

The proposed representation demonstrated a keen ability to deal with the large intra-class variability of handwritten shapes. Yet, the performance of our training-free method is still limited compared to training-based approaches in the literature, highlighting the importance of labeled data to obtain competitive results. In addition, our representation also shows some sensitivity to the degradations that may exist in document images. Another weakness of the proposed method lies in the spherical k-means algorithm, which uses a random initialization of the clusters. This initialization can make the algorithm converge to a local minimum rather than an optimal solution. Furthermore, the proposed representation employs various adjustable parameters

(i.e., size of the patch, number of features, step size, number of layers), the tuning of which can be a long process.

2.5 Conclusions and future works

Two contributions were made to the problem of segmentation-free and training-free word spotting in multi-writers scenario. The first contribution is an unsupervised hierarchical handwriting representation, where the spherical k-means algorithm is used to learn a hierarchy of features. The second contribution is an efficient matching system, which consists of a fast preselection stage and a discriminative re-ranking stage.

Experimental results showed our method to be competitive with state-of-the-art approaches for segmentation-free and training-free word spotting, outperforming these approaches on the GW dataset. Our method could potentially be improved in different ways, for instance by using a more powerful matching system. As in (Almázan *et al.*, 2014b), an Exemplar SVM can be used for this task. Another possible improvement could be to use a query expansion technique, which consists in generating several query word images from the initial query.

CHAPITRE 3

CONVOLUTIONAL PYRAMID OF BIDIRECTIONAL CHARACTER SEQUENCES FOR THE RECOGNITION OF HANDWRITTEN WORDS

Mohamed Mhiri[1], Christian Desrosiers[1], Mohamed Cheriet[1]

[1] Département de génie de la production automatisée, École de Technologie Supérieure, 1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3

Article publié à la revue « Pattern recognition letters », volume 111, Pages 87-93, 2018.

3.1 Introduction

In the last years, the automated recognition of handwritten data has played a key role in the digitization and analysis of historical documents, providing a tremendous help to scholars in various fields. Despite the significant amount of efforts devoted to this problem, handwritten word recognition remains a challenging task due to the large intra-class variability in handwritten shapes and the complexity of modeling and segmenting sequences of overlapping characters. The automated recognition of document images can generally be described as a two-step process, where regions corresponding to individual words are first extracted and then fed to a recognition system (Likforman-Sulem *et al.*, 2007; Louloudis *et al.*, 2009).

Handwriting recognition systems often rely on sequential data models like Hidden Markov Models (HMMs) or Bidirectional Long Short-Term Memory (BLSTM) neural networks. While they have led to significant performance improvements, such models also have important drawbacks (Fink & Plotz, 2007; El-Hajj *et al.*, 2008). Systems based on HMMs typically model words by concatenating many compound HMMs, each one representing single characters. In most cases, these single-character HMMs are defined without considering their context within words (Vinciarelli & Luettin, 2001; Mohamad *et al.*, 2009). Several works have focused on extending standard HMM-based approaches to add contextual information, for instance, by considering surrounding characters (Fink & Plotz, 2007; El-Hajj *et al.*, 2008). However, such strategies may lead to a high number of HMMs, each one with many parameters to learn.

Consequently, they are not suitable when limited training data is available. On the other hand, BLSTMs do not suffer from the context problem of HMMs. This approach uses connectionist temporal classification (CTC) (Graves *et al.*, 2006) to map an input sequence to an output sequence, thus avoiding the need to segment words into separate characters (Graves *et al.*, 2009b). However, as with HMMs, BLSTMs use as input a set of handcrafted features, which may not be able to model the wide variability of handwritten shapes (Youssouf, 2016).

Convolutional neural networks (CNNs) have achieved outstanding performance on a wide range of image analysis tasks (Dolz *et al.*, 2017). Unlike sequential models, CNNs use hierarchical layers of nonlinear processing to learn a high-level and discriminative representation of images (Bengio, 2009; Bengio *et al.*, 2013). This type of neural network is based on three main principles : 1) *local connectivity* – each neuron is only connected to a local region of the input image, thereby modeling local patterns in the image ; 2) *parameter sharing* – units in convolutional layers are repeated in different regions of the input, thus limiting the number of parameters to learn ; 3) *pooling* – parameters are also limited by reducing the representation's spatial size via pooling layers. Pooling operations also make the representation more robust to small translations in the image.

For text recognition (e.g., handwritten or scene text), approaches using CNNs can be grouped in three broad categories :

1. **Feature sequence approaches** (Bluche *et al.*, 2013a; Sun *et al.*, 2016; Baoguang Shi, 2015), where a CNN is first employed to extract sequences of high-level features from the input images, and a BLSTM or HMM model is then used to recognize these sequences as words.
2. **Character recognition approaches** (Alsharif & Pineau, 2013; Jaderberg *et al.*, 2014b), in which the CNN recognizes characters explicitly and an inference step is applied to find the most probable word given the CNN outputs.
3. **Whole-word recognition approaches** (Goodfellow *et al.*, 2013; Jaderberg *et al.*, 2016, 2014a; Poznanski & Wolf, 2016), where a CNN is used to find the word corresponding to an input image, without explicitly recognizing its characters. Generally, these approaches

encode words from a lexicon using a given representation, and the CNN learns the mapping between word images and their representation.

In (Bluche *et al.*, 2013a), a CNN is used to extract sequences of features which are then incorporated within an HMM to recognize word images. It was shown that combining a CNN for feature learning with an HMM for word recognition outperformed HMM models based on hand-crafted features. In (Sun *et al.*, 2016), a convolutional multi-directional recurrent network is presented for handwritten text recognition. This end-to-end trainable model contains a layer called MDirLST, which extracts local context from different directions (right-left, left-right, up-down, down-up). Similarly, (Baoguang Shi, 2015) propose an end-to-end trainable model that combines convolutional layers, several bidirectional LSTM layers and a transcription layer. This model can be applied to arbitrary sized images and achieved remarkable performances in both lexicon-free and lexicon-based text recognition tasks.

In (Alsharif & Pineau, 2013), a character recognition based approach is presented, where word images are initially over-segmented into candidate character regions. The most probable sequence (i.e., word) is then found by combining a CNN-based character recognition technique and a lexicon word recognition method using HMMs. Likewise, (Jaderberg *et al.*, 2014b) propose a character-based method in which a binary CNN classifier is used to detect candidate character regions. Another CNN is then applied across a word image to recognize characters from single and consecutive candidate regions, and the Viterbi algorithm used to find the most probable sequence within a fixed lexicon.

Recently, a few works have considered CNNs in a whole-word recognition setting. In (Goodfellow *et al.*, 2013), a position-sensitive CNN is used to recognize multi-digit numbers from street view images, under the assumption that words have a fixed maximum length. Likewise, (Jaderberg *et al.*, 2016) define whole-word recognition as a classification task across the entire dictionary of potential words, and use CNNs to solve this task. Although it can be applied to handwritten text, this strategy requires many training examples per class, which may not be available in practice. In (Jaderberg *et al.*, 2014a), the same authors proposed two other CNN-based models for whole-word recognition. The first one, called Character

Sequence Encoding (CHAR), uses a single CNN with multiple independent classifiers, each one predicting the character at a specific position in the word. The recognized word is then obtained by finding the lexicon word with the minimum edit distance to the predicted character sequence. In the second model, called Bag-of-N-grams Encoding (NGRAM), words are modeled as an unordered set of N-grams. For instance, the word '`spires`' is encoded as $G_{N=3}$(`spires`) = {`s`,`p`,`i`,`r`,`e`,`s`,`sp`,`pi`,`ir`,`re`,`es`,`spi`,`pir`,`ire`,`res`}. This representation is a $|G_N|$-dimensional binary vector of N-gram occurrences. During inference, given a word image, the CNN output is compared with the N-gram representation of dictionary words and the nearest representation in terms of the Euclidean distance is selected as the recognized word.

In a recent work by (Poznanski & Wolf, 2016), words are represented as an N-gram occurrence vector, which indicates the set of N-grams (i.e., uni-grams, bi-grams and tri-grams) appearing in a word and parts of this word (i.e., spatial regions of the string). This representation, called pyramidal histogram of characters (PHOC), is based on the work of (Almázan *et al.*, 2014a). While Almazan et al. used SVMs from mapping word images encoded as Fisher Vectors of SIFTs descriptors to the label of their N-gram occurrence vector, Poznanski et al. employed a CNN to map word images directly to these vectors. In both these works, a Canonical Correlation Analysis (CCA) is used for inference, where a common subspace is learned for the predicted vectors (i.e., the outputs of the layer below the prediction layers in (Poznanski & Wolf, 2016)) and the PHOC representations. Words are then recognized via a simple nearest-neighbor search in the learned common subspace.

In this paper, we propose a novel method using CNNs for whole-word recognition in handwritten texts. Unlike character-based approaches (Alsharif & Pineau, 2013; Jaderberg *et al.*, 2014b), our method does not require the explicit segmentation of characters, a particularly difficult task for handwritten words. Moreover, as opposed to sequential models (Vinciarelli & Luettin, 2001; Mohamad *et al.*, 2009; Alsharif & Pineau, 2013; Jaderberg *et al.*, 2014b), the proposed method does not use handcrafted features, and can learn automatically a suitable representation from the training data. The main contribution of this work is a novel word representation, called Pyramid of Bidirectional Character Sequences (PBCS), which encodes both

forward and backward sub-sequences of characters in a hierarchical manner. Using this representation, a single CNN network can learn the distribution of character sub-sequences in the data, and transfer this knowledge across words containing the same sub-sequences (e.g., the sub-sequence '`to`' is a word and also a sub-part of the word '`together`'). As a result, the proposed method may recognize words with fewer training examples per class.

Similar to the PHOC-based representation (Almázan *et al.*, 2014a; Poznanski & Wolf, 2016), our PBCS representation is also a binary vector modeling the occurrence of characters in sub-parts of the input word. However, because their representation is a bag of N-grams, it models the presence of these N-grams in specific sub-words, but not the order in which they appear. Moreover, since a different CNN output is required for each possible N-gram of every sub-word, to avoid an exponential increase of CNN parameters, character sequences are limited to tri-grams. In contrast, our PBCS representation can encode character sequences of any length, in both forward and backward directions, and the number of CNN parameters increases only linearly with respect to this length. Consequently, the proposed representation is much more compact than PHOC, with $\sim$2K features compared to $\sim$10K for PHOC using the same alphabet of 52 characters. This decreases the number of parameters to learn, thus reducing computation times and chances of overfitting. The proposed framework also differs from the strategy of (Jaderberg *et al.*, 2016), where the CNN output corresponds to word classes. As mentioned above, mapping word images to character sub-sequences, instead of whole-word labels, allows transferring knowledge across words with common sub-sequences.

The rest of the paper is organized as follows. In Section 3.2, we present our proposed CNN approach for the whole-word recognition of handwritten text. Section 3.3 then evaluates the usefulness of this approach on the well-known IAM and RIMES databases, showing state-of-the-art performance compared to recently proposed methods for this task. Finally, we conclude by summarizing the contributions and results of this work.

3.2 The proposed method

The proposed whole-word recognition framework is shown in Figure 3.1. This framework is composed of two stages. Given an image I of a handwritten word, we first use a CNN to learn the mapping from I to its PBCS representation. As described in Section 3.3.2, this CNN is composed of eight convolutional layers with non-linear activation units, followed by three fully-connected layers, and a softmax normalization layer at the end of the network. The output of this CNN is a sequence of probabilities, corresponding to the possible values of each PBCS element. In the second stage, an inference strategy is used to find the most probable word in a lexicon for this output.

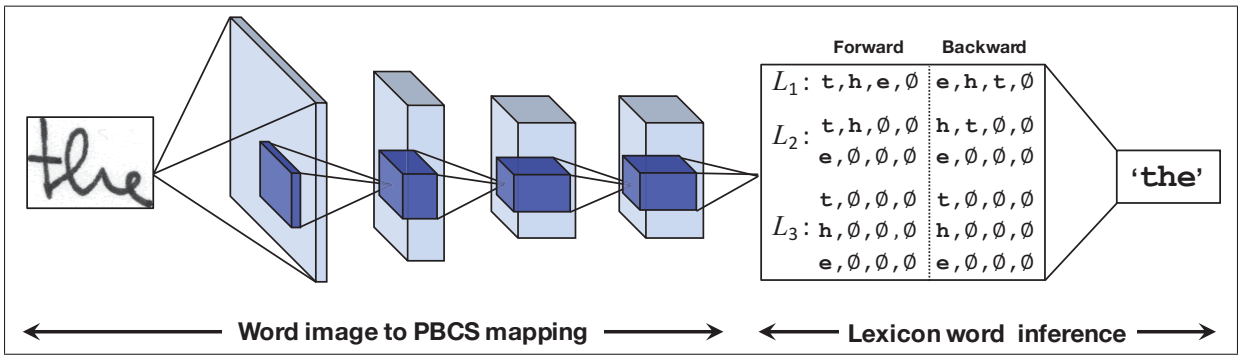


Figure 3.1 The proposed framework. A deep single CNN first outputs a sequence of character probabilities (Z independent character classifiers) for an input word image. An inference technique is then used to find the most probable lexicon word based on these probability values.

To use our method, two problems must first be tackled : 1) the non-fixed size of the input image, and 2) the variable length (in characters) of the word it represents. For instance, the images in Figure 3.2(a) have different sizes and represent words containing a different number of characters. While convolution operations may be applied to images of arbitrary size, fully-connected layers require a fixed-size input. Likewise, the CNN output vector must be of fixed length, regardless of the actual number of characters in the input word.

In the following two sub-sections, we describe how the proposed framework deals with the non-fixed size of input images and the variable length of input words. We then present the strategies

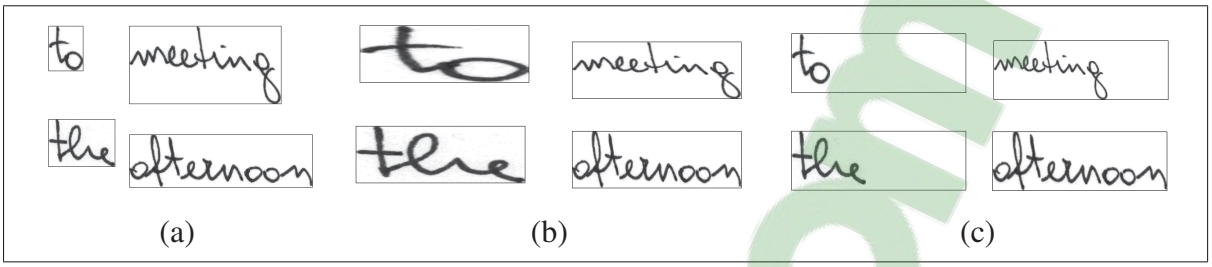


Figure 3.2 The proposed resizing technique. (a) Word images of different size and their resized version using (b) standard resizing or (c) the proposed technique.

employed to normalize the CNN outputs into probabilities and converting these probabilities into a word from a known lexicon.

3.2.1 Non-fixed size of input images

A simple solution to this problem is to resize the input image to a pre-determined size. However, this resizing process may affect the aspect ratio of images (i.e., the ratio between image width and height). In addition, this process can represent the same characters with different scales, making harder the task of recognizing these characters. To avoid this problem, we propose a more sophisticated resizing technique, which right-pads word images with white space until reaching a fixed size. Figure 4.3 shows two examples of word images, and the result of resizing these words using the standard approach and the proposed technique. It can be seen that the proposed technique better preserves the aspect ratio of these images.

3.2.2 Pyramid of bidirectional character sequences (PBCS)

While words may have a variable number of characters, the output of the CNN must be a fixed-length vector. A simple strategy to address this problem is to limit the word representation to N characters, shorter words being padded with a special void character (i.e., $\emptyset$). For instance, using $N=4$, the word '`cat`' would be represented as {`c`, `a`, `t`, $\emptyset$}. However, due to the heavy-

tailed distribution of word lengths, shorter words are more frequent than longer ones. This leads to an over-representation of the first characters and, thus, a poor recognition of longer words.

To overcome these problems, we propose a novel word representation that builds a bidirectional pyramid of character sub-sequences. The pyramid is composed of L levels, with the l-th level dividing the word into l even-sized sub-words. Let W be the length of the input word. If W is dividable by l, then each sub-word will contain exactly W/l characters, otherwise there will be $l-1$ sub-words of length $\lceil W/l \rceil$ and another sub-word of length $W-(l-1)\lceil W/l \rceil$. These sub-words are encoded as sequences of N characters, truncating sub-words exceeding this length and padding with the void character $\emptyset$ those having less than N characters. To make the representation more robust, sub-words are encoded in both directions, left-to-right and right-to-left, by considering their first N and last N characters.

The final representation corresponds to the concatenation of sequences from all three levels. Using an alphabet of K possible values for each character (plus the void character), a pyramid of L levels, and representing the N first and last characters, the proposed representation has a total of $(K+1) \times N \times 2 \times \sum_{l=1}^{L} l$ features. For instance, if $N=4$, $L=3$ and $K=26$ (i.e., the number of letters in the Latin alphabet), the PBCS representation is a vector of size $(26+1) \times 4 \times 2 \times (1+2+3) = 1296$.

Table 3.1 shows the proposed pyramid of bidirectional character sequences (PBCS) representation of the words '`the`' and '`there`', using $L=3$ and $N=4$. Despite these words having different numbers of characters and sharing a common sub-sequence '`the`', their representations have the same length but are quite different. Moreover, the benefit of having bidirectional character sequences can be seen in the first level of the word '`there`', where the forward and backward sequences differ (i.e., the backward sequence is not the reverse of the forward one) due to the length of the word exceeding N. The proposed representation can thus model the beginning and ending of words, regardless of their length. In our experiments, we show that words can be effectively recognized considering only these short sequences.

Tableau 3.1 PBCS representation of the words ‘the’ and ‘there’ using $N=4$ and $L=3$. For each level, the first row gives the character sub-sequences used for encoding. Columns correspond to taking the N first and N last characters of these sub-sequences, padding with the void character ∅.

Word	‘**the**’		‘**there**’	
1st level	‘the’		‘there’	
	t, h, e, ∅	e, h, t, ∅	t, h, e, r	e, r, e, h
2nd level	‘th’ + ‘e’		‘the’ + ‘re’	
	t, h, ∅, ∅	h, t, ∅, ∅	t, h, e, ∅	e, h, t, ∅
	e, ∅, ∅, ∅	e, ∅, ∅, ∅	r, e, ∅, ∅	e, r, ∅, ∅
3rd level	‘t’ + ‘h’ + ‘e’		‘th’ + ‘er’ + ‘e’	
	t, ∅, ∅, ∅	t, ∅, ∅, ∅	t, h, ∅, ∅	h, t, ∅, ∅
	h, ∅, ∅, ∅	h, ∅, ∅, ∅	e, r, ∅, ∅	r, e, ∅, ∅
	e, ∅, ∅, ∅	e, ∅, ∅, ∅	e, ∅, ∅, ∅	e, ∅, ∅, ∅

We note that our PBCS representation encodes a word with redundant information (i.e., character sequences overlap), in a way that is similar to lifting techniques (Candes *et al.*, 2015) for encoding matrices. This redundancy helps discriminate between words in the inference step, at the cost of a greater representation size.

3.2.3 Output layer normalization

In standard CNNs for classification, the output vector is normalized such that the sum of elements in this vector equals one (as in a probability distribution). This is typically achieved via a softmax function applied over the entire output vector. However, in our case, the normalization should be applied separately to each element in the PBCS representation. This allows possibly overlapping sub-sequences of a word to be modeled and detected jointly by the CNN. Let y_{ij} be the network's output corresponding to i-th position in the PBCS representation and the j-th possible character value. This normalization is carried out as

$$o_{ij} = \frac{\exp(y_{ij})}{\sum_{j=1}^{K+1}\exp(y_{ij})}, \quad i = 1,\ldots,M, \tag{3.1}$$

where $M = N \times 2 \times \sum_{l=1}^{L} l$.

We use multi-class cross-entropy as loss function to optimize the network parameters. Let T be the PBCS representation of a training word, where $t_{ij} = 1$ if this representation has a character j in position i, else $t_{ij} = 0$. For a single input word image, this loss function corresponds to

$$\text{loss}(O,T) \;=\; -\sum_{i=1}^{M}\sum_{j=1}^{K+1} t_{ij}\cdot\log(o_{ij}). \tag{3.2}$$

In training, this loss is computed over sets of words forming small batches.

3.2.4 Mapping CNN outputs to lexicon words

During testing, an additional step is required to convert the CNN output probability values O to a word from a known lexicon (or *dictionary*) $\mathscr{D}$. Toward this goal, we consider the PBCS representation of each lexicon word w as a set of random variables $T^w = \{t_{ij}^w\}$, $i = 1,\ldots,M$, $j = 1,\ldots,K+1$, and maximize the posterior probability :

$$argmax_{w\in\mathscr{D}}\ \log P(T^w \,|\, O). \tag{3.3}$$

Assuming that the elements of T^w are conditionally independent given O, this inference problem becomes

$$argmax_{w\in\mathscr{D}}\ \sum_{i=1}^{M}\sum_{j=1}^{K+1} t_{ij}^w\cdot\log(o_{ij}) \tag{3.4}$$

which corresponds to negative cross-entropy. After calculating the logarithm of outputs O once, the score of all lexicon words can be computed in a single efficient operation corresponding to the product between a sparse matrix and a vector.

3.3 Experiments

In this section, we evaluate the proposed whole-word recognition approach on the IAM Off-line (Marti & Bunke, 2002) and RIMES (Grosicki *et al.*, 2009) databases. We start by describing these two databases, and then present our CNN architecture and techniques used for training. Finally, we compare the performance of our approach against recent word recognition methods.

3.3.1 Databases

Two separate databases were used in our experiments :

- The Off-line IAM database (Marti & Bunke, 2002) is a large set of English texts comprised of 115,320 labeled words from 657 different writers. In our experiments, we used a subset of 96,456 labeled word images, remaining ones discarded due to an incorrect segmentation. While the database contains four pre-defined subsets of examples, a training set, two distinct validation sets and a testing set, we combined all examples from the pre-defined training and validation sets 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 (Grosicki *et al.*, 2009) 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 testing sets. As in the IAM database, we combine all examples from the training and validation sets into a single training set of 51,737 words images corresponding to 4637 distinct words. The testing set contains 7464 word images representing a total of 1509 distinct words. As lexicon, we considered words from both training and test sets, giving a vocabulary of 4989 words. Ignoring letter case, words in the RIMES database contain $K = 53$ different characters : `'-/0123456789abcdefghijklmnopqrstuvwxyz-°²àâçèéêëîïôùû`

3.3.2 Network architecture

Inspired by the VGG-16 network (Simonyan & Zisserman, 2015), which has shown that depth plays an important role the network's performance, the CNN architecture used in our experiments contains a total of 15 layers (about 35 million parameters). As detailed in Table 3.2, this

architecture has 8 convolutional layers, 3 max-pooling layers, 3 fully-connected layers, and 1 normalization layer. All hidden (i.e., convolutional and fully-connected) layers use rectified linear units (ReLU) (Bengio *et al.*, 2013). Moreover, as recommended in (Ioffe & Szegedy, 2015), batch normalization is applied before each activation layer. The last fully-connected layer is followed by the normalization operation described in Section 3.2.3. The input word images are provided as grey-level images of varying dimensions. Using the technique presented in Section 3.2.1, these word images were resized to a common size of 40×170 pixels. No binarization technique is applied to the images, nor other pre-processing steps such as the correction of slanted characters/skewed images.

Tableau 3.2 The CNN architecture used in our experiments.

Layer (type)	**Input shape**	**Output shape**
Conv2D (with padding)	40×170	$64 \times 40 \times 170$
Conv2D (with padding)	$64 \times 40 \times 170$	$64 \times 40 \times 170$
Max-pooling	$64 \times 40 \times 170$	$64 \times 20 \times 85$
Conv2D (with padding)	$64 \times 20 \times 85$	$128 \times 20 \times 85$
Conv2D (with padding)	$128 \times 20 \times 85$	$128 \times 20 \times 85$
Max-pooling	$128 \times 20 \times 85$	$128 \times 10 \times 42$
Conv2D (with padding)	$128 \times 10 \times 42$	$256 \times 10 \times 42$
Conv2D (with padding)	$256 \times 10 \times 42$	$256 \times 10 \times 42$
Max-pooling	$256 \times 10 \times 42$	$256 \times 5 \times 21$
Conv2D (with padding)	$256 \times 5 \times 21$	$512 \times 5 \times 21$
Conv2D (without padding)	$512 \times 5 \times 21$	$512 \times 1 \times 17$
Fully-connected	8704×1	2000×1
Fully-connected	2000×1	2000×1
Fully-connected	2000×1	2544×1 (IAM) 2592×1 (RIMES)

3.3.3 Implementation details

The training data was augmented using random rotations ranging from 1 to 10 degrees, random translations with vertical and horizontal shift ranges of $0.05 \times$ height and $0.05 \times$ width, and random zooms with factors between 0.9 and 1.1. During testing, test images were used without any modification.

For training, we used the Stochastic Gradient Descent optimization technique with an initial learning rate of 0.01, a momentum of 0.9 and a weight decay of 10^{-6}. Furthermore, to reduce chances of overfitting, dropout (Srivastava *et al.*, 2014) was employed in all convolutional layers with a dropout rate of 25%, and in the fully-connected layers with a dropout rate of 50%. Since void characters are more frequent, to avoid problems resulting from unbalanced data, a class weight of 0.001 was assigned to outputs representing such characters. In contrast, a class weight of 1 was given to the outputs of all other characters.

All the experiments were performed on a GeForce GTX 960 GPU with a clock speed of 1190 MHz. Training the network takes approximately 52 hours for the IAM database and about 40 hours for the RIMES database, using 50 epochs with batch size of 64.

3.3.4 Results

Two standard metrics were used to measure the performance of our word recognition method : character error rate (CER) and word error rate (WER). 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. On the other hand, the WER is the percentage of words that are wrongly transcribed.

3.3.4.1 Impact of representation parameters

Tableau 3.3 Word error rates obtained by our approach with different numbers of characters N and numbers of levels L for the PBCS representation.

Database	**RIMES**			**IAM**		
	$L=1$	$L=2$	$L=3$	$L=1$	$L=2$	$L=3$
$N=2$	36.59%	8.09%	7.54%	30.19%	12.93%	9.83%
$N=3$	11.58%	7.37%	7.45%	13.02%	**9.41**%	9.54%
$N=4$	8.74%	**7.30**%	7.60%	10.33%	9.57%	9.69%

Table 3.3 gives the WER obtained by our approach, using different numbers of characters N and numbers of levels L for the PBCS representation. It can be seen that, regardless the number of characters used, representations having two or three levels outperform those with a single level. Differences are most important for representations with $N=2$ characters. For instance, in the RIMES database, using a three-level representation yields a WER improvement of 29.05% over a single level representation. Comparing two-level and three-level representations, we observe similar performances with the lowest WER values obtained for $L=2$. Overall, these results demonstrate the usefulness of a multi-level representation, although benefits are less important beyond three levels.

To further analyze these results, we show in Figure 3.3 the accuracy (i.e., percentage of correct character predictions) for the different characters in the representation defined with $N=4$ and $L=3$. For each level (L), sub-word (S) and direction (fwd/bwd), the accuracy follows the position of characters, early ones (e.g., position 1 or 2) having a smaller WER than later ones (e.g., position 3 or 4). This is likely due to the accumulation of errors when decoding sequences : an error at a given position often induces errors in following ones. With respect to sub-word position and direction, a slightly higher accuracy is found for later sub-words and for backward sequences. This could be explained in part by the skewed distribution of characters at the end of a word, where the same characters are often found (e.g., the void character used for padding).

The advantages of using a multi-level and bidirectional representation are further analyzed in Table 3.4, which gives the WER obtained by considering only the $N=4$ first characters, the first level (i.e., the $N=4$ first and last characters of the input word), or the whole PBCS representation. We observe that using only the first four characters of the representation is not sufficient to distinguish between words, due to common word prefixes, stems, etc. in the language. For instance, the words ‘part’ and ‘parts’ cannot be differentiated using only these characters. However, a more discriminative representation is obtained by adding the last four characters. Thus, combining both the first and last four characters reduces the WER by over 27% in the RIMES database and over 16% in the IAM database. Moreover, considering all three levels of the pyramid can further improve results by about 1% in both databases.

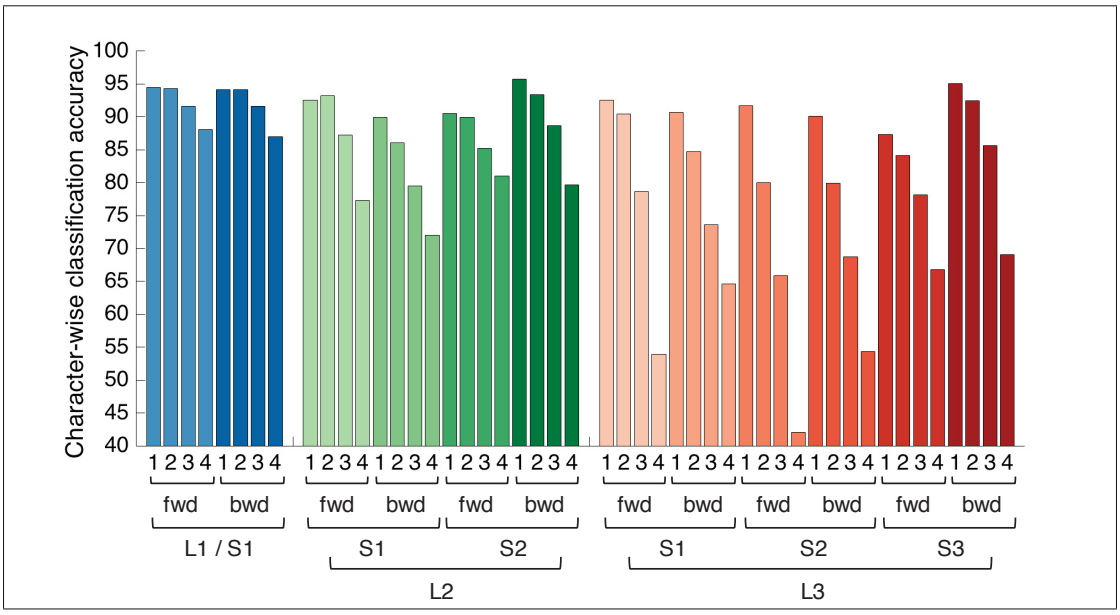


Figure 3.3 The character-wise classification accuracy for different positions of the PBCS representation defined with $N=4$ and $L=3$, on test examples of the IAM database.

Tableau 3.4 Word error rates obtained using different parts of the PBCS representation defined with $N=4$ and $L=3$.

Subpart	**RIMES**	**IAM**
$N=4$ first characters	36.41%	26.45%
$N=4$ first and last characters (first level)	8.74%	10.33%
PBCS representation (three levels)	**7.60%**	**9.69%**

3.3.4.2 Impact of word length

Figure 4.6 compares the word error rates and number of incorrect word recognitions corresponding to different lengths of words in the IAM database. Although more errors occur for small words (e.g., wrongly transcribed single character words account for about 29% of the total errors), our approach actually performs better with short words than long ones. In particular, an important increase in error rate is observed for words with 12 or more characters, suggesting that the number of levels and the length of sub-words in the representation might be insufficient to model these longer words. Moreover, inspecting errors on single-character words

reveals that most of them are caused by similar-looking punctuation characters (i.e., '.' versus ',' or ' :' versus ' ;'). These errors could possibly be eliminated by applying a grammar-based model in post-processing.

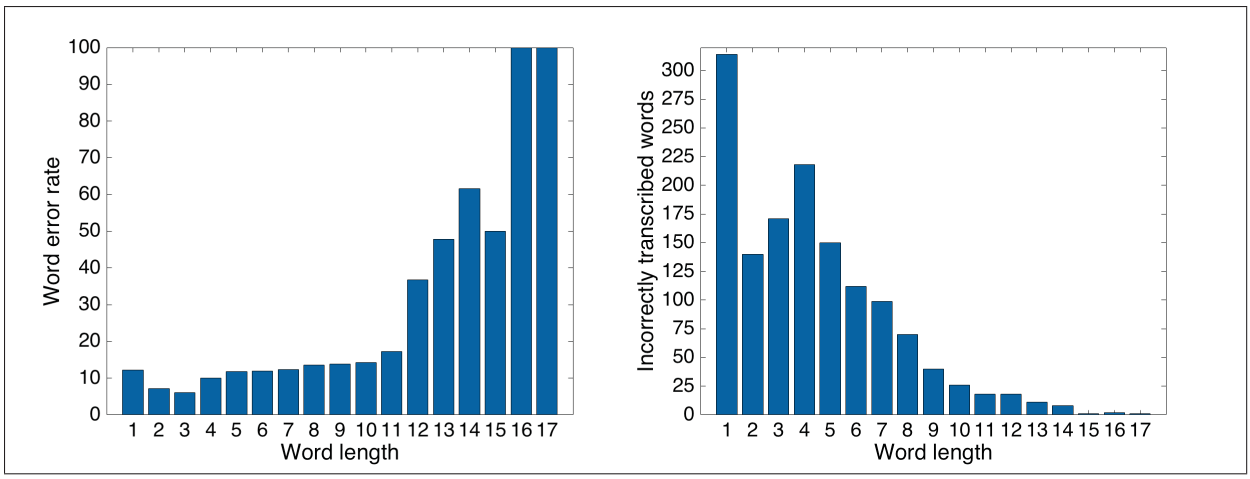


Figure 3.4 The word error rates and number of incorrect word recognitions corresponding to different lengths of words in the IAM database.

In another analysis, we measured whether the proposed approach could predict the length of test words in the IAM database (i.e., the percentage of the transcribed words have the same length as the ground truth words). We found that the correct length was predicted for 96% of these words, confirming the ability of our representation to effectively model character sequences.

3.3.4.3 Comparison with the state-of-the-art

Table 4.8 compares the performance of our approach against that of recent methods for word recognition in handwritten texts. For all methods, the WER and CER was measured on test examples of the RIMES and IAM databases, using the words in the training and testing sets as lexicon. We report results obtained using only the pre-defined training set, or the combined training and validation sets (denoted as training set* in the table). Moreover, we investigated an augmentation technique based on the elastic transform (Simard *et al.*, 2003) to generate ad-

Tableau 3.5 Word and character error rates obtained by our PBCS representation defined with $N=4$ and $L=3$ and recent word recognition methods on test examples of the RIMES and IAM databases.

Method	**RIMES**		**IAM**	
	WER	CER	WER	CER
Bianne-Bernard et al. (Bianne-Bernard *et al.*, 2011) (HMM)	14.7%	–	21.9%	–
Espana-Boquera et al. (Espana-Boquera *et al.*, 2011) (HMM)	16.8%	–	21.2%	9.1%
Bluche et al. (Bluche *et al.*, 2013b) (LSTM)	9.2%	–	20.5%	–
Bluche et al. (Bluche *et al.*, 2014) (LSTM)	11.8%	3.7%	11.9%	4.9%
Pham et al. (Pham *et al.*, 2013) (LSTM)	12.3%	3.3%	13.6%	5.1%
Menasri et al. (Bianne-Bernard & Kermorvant, 2012) (HMM + LSTM)	4.82%	–	–	–
Almazan et al. (Almázan *et al.*, 2014a) (PHOC)	–	–	20.01%	11.27%
Poznanski et Wolf (Poznanski & Wolf, 2016) (full CNN-N-gram)	**3.90%**	1.90%	**6.45%**	3.44%
Poznanski et Wolf (Poznanski & Wolf, 2016) (without using CCA)	5.17%	3.43%	8.83%	5.93%
Ours (PBCS + training set)	8.10%	4.51%	10.26%	6.82%
Ours (PBCS + training set*)	7.60%	3.72%	9.69%	6.30%
Ours (PBCS + training set* + elastic transform)	6.22%	3.28%	8.83%	5.95%

Note : training set* contains both training and validation examples of the corresponding database.

ditional word images during training. In this technique, the intensity x of a pixel is changed to $f(x) = x + \alpha x(x-1)(x-\beta)$. Parameter α is randomly sampled in $[-2,2]$ and β in $[0.4,0.6]$. This augmentation technique makes the CNN model less sensitive to differences in pixel intensities (i.e., contrast), thereby avoiding the need for image binarization. Results obtained using this technique are shown in the last row of Table 4.8. We observe a significant improvement in performance for both databases, suggesting that additional efforts on data augmentation could yield even better results.

With respect to other tested methods, our approach outperforms largely the method of (Bianne-Bernard *et al.*, 2011), which combines three HMM-based recognizers to consider dynamic and contextual information of characters. Similarly, our approach shows better performances than the HMM-based method of (Espana-Boquera *et al.*, 2011), however, a direct comparison is not possible since handwritten line images were used in this work. Furthermore, the proposed CNN approach compares favorably to recent methods (Bluche *et al.*, 2013b, 2014; Pham *et al.*, 2013) based on LSTMs, outperforming all but one of these methods. The method in (Bianne-Bernard & Kermorvant, 2012) uses a weighted combination of seven different models (one hybrid MLP-HMM, two tandem GMM-HMMs, four MDLSTMs), making it computationally

expensive and hard to adapt to new data. Although we failed to outperform the results of (Poznanski & Wolf, 2016), our approach offers significant advantages in terms of computational efficiency. Hence, the method of Pozanski et al. applies a test-side augmentation technique that generates 37 variations of the test word image, and computes for each of them a representation using a 19 multi-layer CNN. These 37 representations are then averaged to obtain the final representation. In contrast, our model uses only the original test word image, and computes its PBCS representation in 9.3 ms, on average.

Tableau 3.6 Top-3 and Top-10 word error rate.

	RIMES	**IAM**
Top 1	6.22%	8.83%
Top 3	3.24%	3.90%
Top 10	1.42%	1.69%
Top 20	0.83%	0.99%

Table 3.6 presents the top-3, top-10 and top-20 word error rates (i.e., the frequency at which the 3, 10 and 20 most probable dictionary words predicted by our CNN do not include the actual test word to predict). We see that our approach achieves a top-20 WER less than 1% in both the IAM database (lexicon of 11,118 words) and the RIMES database (lexicon of 4989 words). Our approach could therefore be used to pre-select a shortlist of candidate words from the dictionary, before applying a more language-oriented model.

3.4 Conclusions

We presented a novel approach for handwritten word recognition using a deep convolutional neural network. In contrast to most methods for this task, the proposed approach works without the need for character segmentation or handcrafted features. Given a training set of word images, our approach learns a deep hierarchy of features corresponding to bidirectional character sequences, which can represent the wide variability of handwritten shapes. By mapping word images to this hierarchical representation, it implicitly learns the distribution of character sequences and their location in the data. This information can be transferred across words

containing these same sequences, thus limiting the number of examples required for training. Experiments on the IAM and RIMES databases demonstrate that our approach can effectively recognize sequences of characters without the complexity of modeling and segmenting overlapping characters. Compared to recent methods, our approach provides competitive results and a high computational efficiency.

CHAPITRE 4

WORD SPOTTING AND RECOGNITION VIA A JOINT DEEP EMBEDDING OF IMAGE AND TEXT

Mohamed Mhiri[1], Christian Desrosiers[1], Mohamed Cheriet[1]

[1] Département de génie de la production automatisée, École de Technologie Supérieure, 1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3

Article soumis à la revue « Pattern recognition », accepté avec révision.

4.1 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 Giotis *et al.* (2017); Manmatha *et al.* (1996); Rodriguez-Serrano & Perronnin (2012). 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.

Following previous studies such as Almázan *et al.* (2014a), 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 Bissacco *et al.* (2013); Manmatha & Rothfeder (2005); Neumann & Matas (2013). 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 Almázan *et al.*

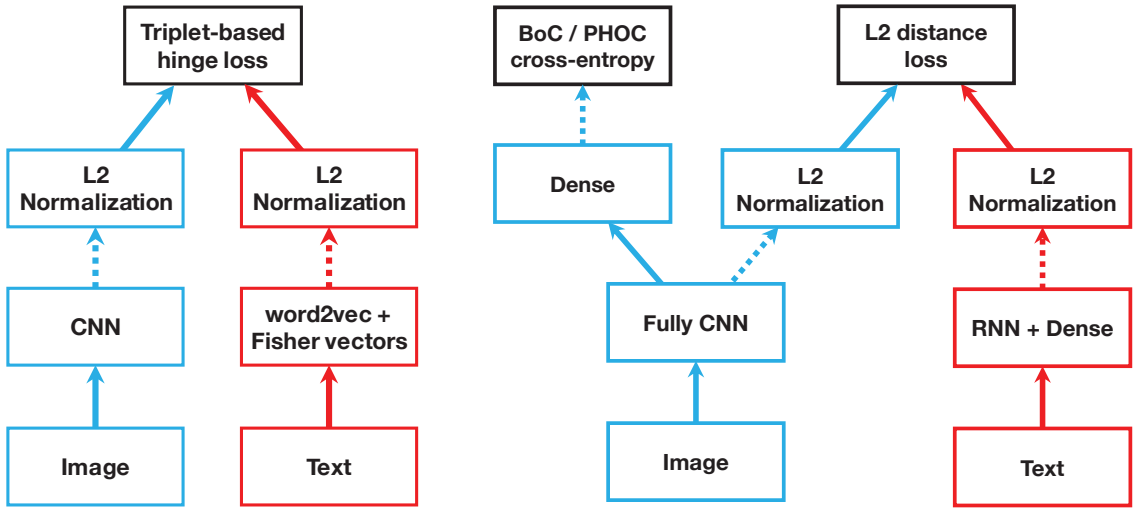
(2014a), limits the recognition to dictionary words. However, as shown in our experiments, it also has the important benefit of allowing cross-database / language transfer of learned information, using a different lexicon in testing than in learning.

Although word spotting and recognition are closely related, most existing approaches consider these tasks separately (Graves *et al.*, 2009b; Rodriguez-Serrano & Perronnin, 2012). 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 Almázan *et al.* (2014a), 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 Wang *et al.* (2016), 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. 4.1 (a), converts individual words in a sentence to word2vec vectors (Mikolov *et al.*, 2013) 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 Figure 4.1 (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 represen-

tation. 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.



a) Embedding model of Wang *et al.* (2016) b) Proposed architecture

Figure 4.1 A comparison of the image-text joint embedding approach in Wang *et al.* (2016) 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).

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 Almázan *et al.* (2014a) and Wang *et al.* (2016), 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 4.2 gives an overview of related work on word spotting and word recognition. In Section 4.3, 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 4.4, 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 5.4 concludes the paper with a summary of main contributions and results.

4.2 Related work

Over the years, various representations have been proposed for encoding word images, including those based on dictionary learning (Mhiri *et al.*, 2018), Scale-Invariant Feature Transform (SIFT) descriptors (Rusinol *et al.*, 2015; Konidaris *et al.*, 2016), Histogram of Oriented Gradients (HOG) descriptors, and Fisher Vectors (Almázan *et al.*, 2014a; **?**). However, recent works have shown the advantages of inferring the representation in a data-driven manner using deep learning (Bengio, 2009; Bengio *et al.*, 2013). In Jaderberg *et al.* (2016), a CNN is employed to detect and recognize words in natural images. The work in Poznanski & Wolf (2016) 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-/trigrams called Pyramidal Histogram of Characters (PHOC) (Almázan *et al.*, 2014a). 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 Sudholt & Fink (2016), 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 Sudholt & Fink (2017), they modified their PHOCNet architecture to process arbitrarily-sized images by adding a spatial pooling layer He *et al.* (2015) after the last convolutional layer. In Sudholt & Fink (2018), 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. Wilkinson *et al.* (2017) 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 Almázan *et al.* (2014a); Wang *et al.* (2016). Inspired by this, Krishnan et al. Krishnan *et al.* (2016) proposed using a pre-trained CNN called HWNet Krishnan & Jawahar (2016) 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. Almázan *et al.* (2014a), 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 L. Gomez & Karatzas (2018), 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 Levenshtein (2012) of word pairs Levenshtein (2012) 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 (Vinciarelli & Luettin, 2001; Mohamad *et al.*, 2009; Roy *et al.*, 2016). Several works have focused on extending standard HMM-based models to add contextual information, for instance, by considering surrounding characters (Fink & Plotz, 2007; El-Hajj *et al.*, 2008). However, such strategies may lead 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) (Graves *et al.*, 2006, 2009b; Wöllmer *et al.*, 2013) 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.

4.3 Methodology

The proposed network architecture is illustrated in Figure 4.2. 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 fully-CNN is used for mapping word images to a set of high-level features that are robust to various transformations like small translations (Bengio *et al.*, 2013; Dolz *et al.*, 2017). 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 (Chung *et al.*, 2014), which learn the dependencies between ch

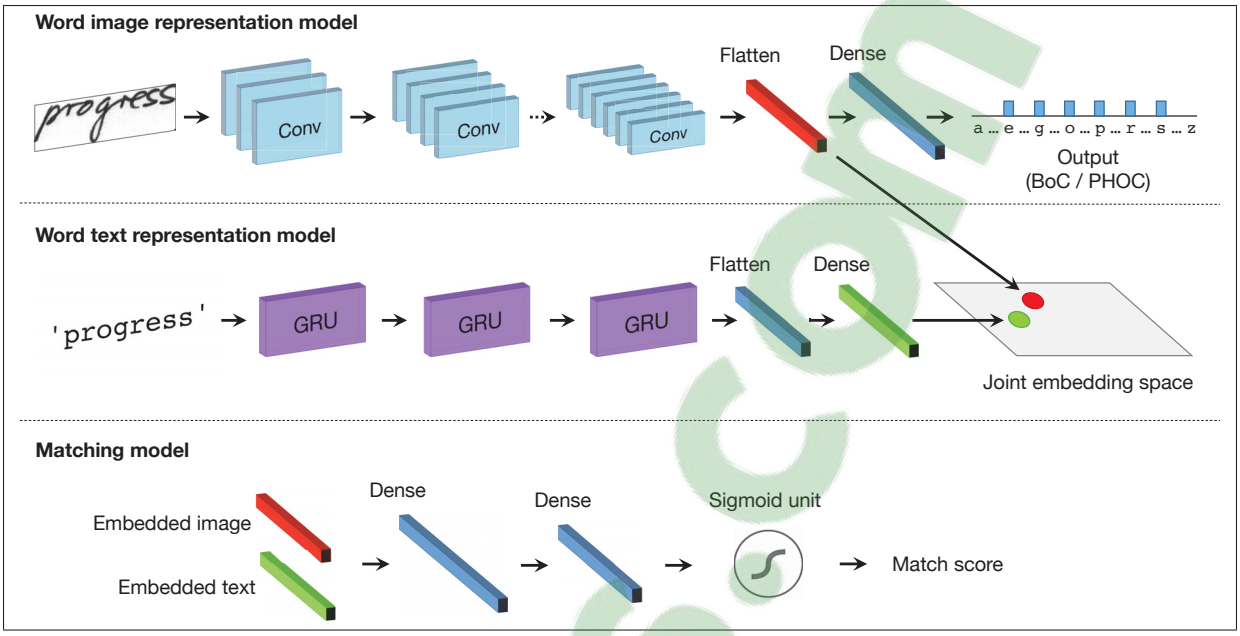


Figure 4.2 The proposed framework that learns a joint embedding of word images and texts to address the word spotting and recognition tasks.

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 text, 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.

4.3.1 Problem formulation

Let $\mathbf{X} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$ 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 (Hadsell *et al.*, 2006) that also penalizes non-matching image-text pairs whose embedding distance is below a given threshold. Alternatively, one can use a ranking loss (Wang *et al.*, 2016) 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 θ_I and θ_T the parameters of embedding functions f_I and f_T, respectively. We define the embedding problem as minimizing loss function

$$J(\theta_I, \theta_T; \mathbf{X}) \;=\; \sum_{i=1}^{N} J_{\text{dist}}\big(f_I(\mathbf{x}_i; \theta_I), f_T(\mathbf{y}_i; \theta_T)\big) \;+\; J_{\text{struct}}\big(f_I(\mathbf{x}_i; \theta_I), \mathbf{s}_i\big), \tag{4.1}$$

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 $f_I(\mathbf{x}_i; \theta_I)$ and $f_T(\mathbf{y}_i; \theta_T)$, 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 :

$$J_{\text{dist}}\big(\mathbf{f}_I, \mathbf{f}_T\big) \;=\; \frac{1}{M} \big\| \mathbf{f}_I - \mathbf{f}_T \big\|_2^2. \tag{4.2}$$

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

Although any fixed-size representation could be considered to model word structure, we use a simple bag-of-characters encoding. Let K be the number of characters in the alphabet. Training word i is encoded as a binary vector $\mathbf{s}$ of size K, whose k-th bit equals 1 if the corresponding character of the alphabet is in the word, else the bit is 0. We first learn a parametric model $g_S(\mathbf{f}_I;\theta_S)$ which outputs, for each character k of the alphabet, the probability p_k that the word represented by $\mathbf{f}_I$ contains this character. Then, J_{struct} is defined as the mean cross-entropy between the predicted probabilities and character occurrences in $\mathbf{s}$:

$$J_{\text{struct}}(\mathbf{f},\mathbf{s}) \;=\; -\frac{1}{K}\sum_{k=1}^{K} s_k \log p_k + (1-s_k)\log(1-p_k). \tag{4.3}$$

As described in the following sections, parameters θ_I, θ_T and θ_S are learned jointly in an end-to-end fashion.

4.3.2 Word image embedding

Table 4.1 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) (Bengio *et al.*, 2013). Moreover, as recommended in (Ioffe & Szegedy, 2015), batch normalization is applied before each activation layer. After the last convolution layer, feature 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 bag-of-characters representation. The output layer has K sigmoid units, each one giving the occurrence probability of a character in the alphabet (e.g., $K = 52$ for the IAM database, and $K = 53$ for the RIEMS database). Since words may have multiple characters, the proposed architecture has no softmax layer before the output.

As mentioned in the previous section, the bag-of-characters (BoC) representation is used to inject character-level information in the embedding process. Unlike PHOC, this simple re-

Tableau 4.1 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.*

presentation does not encode information on the location of characters. This design choice is motivated by the fact that fully-CNNs implicitly capture spatial information by applying convolution filters locally. In prelimnary experiments, we found no significant advantage of using PHOC instead of BoC as final representation.

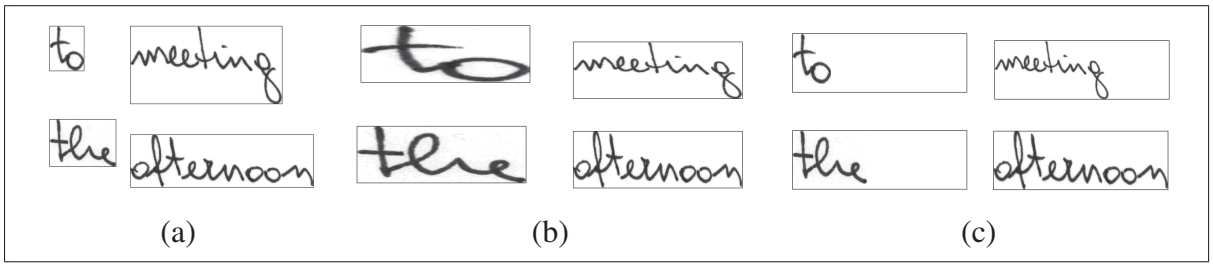


Figure 4.3 Word image resizing. (a) Word images of different size and their resized version using (b) the standard resizing technique, and (c) the proposed technique.

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 4.1, which right-pads word images with white space until reaching a fixed size of 40×170 pixels. Figure 4.3 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.

Algorithme 4.1 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**
    $\mathbf{I}_R(i,j) := \mathbf{I}(i,j),\ \ i = 1,\ldots,h,\ \ j = 1,\ldots,w;$
else
    if $h_R/h \leq w_R/w$ **then**
        $\mathbf{T} := \text{resize}\big(\mathbf{I}, [h_R, w \times h_R/h]\big);$
        $\mathbf{I}_R\big(i,j\big) := \mathbf{T}(i,j),\ \ i = 1,\ldots,h_R,\ \ j = 1,\ldots,w \times h_R/h;$
    else
        $\mathbf{I}_R := \text{resize}\big(\mathbf{I}, [h_R, w_R]\big);$
    end
end
return $\mathbf{I}_R$

4.3.3 Word text embedding

To embed the character string of a word, we use the neural network architecture given in Table 4.2. 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 :

$$\mathbf{z}_t = \sigma_g\big(\mathbf{W}_z\mathbf{x}_t + \mathbf{U}_z\mathbf{h}_{t-1} + \mathbf{b}_z\big) \tag{4.4}$$

$$\mathbf{r}_t = \sigma_g\big(\mathbf{W}_r\mathbf{x}_t + \mathbf{U}_r\mathbf{h}_{t-1} + \mathbf{b}_r\big) \tag{4.5}$$

$$\mathbf{h}_t = \mathbf{z}_t \circ \mathbf{h}_{t-1} + (1-\mathbf{z}_t) \circ \sigma_h\big(\mathbf{W}_h\mathbf{x}_t + \mathbf{U}_h(\mathbf{r}_t \circ \mathbf{h}_{t-1}) + \mathbf{b}_h\big) \tag{4.6}$$

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.

Tableau 4.2 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 units, **our word text representation**)	**2176**

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.

4.3.4 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 Figure 4.2) 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 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.

Tableau 4.3 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**

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 4.3 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 4.2, 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.

Algorithme 4.2 The training algorithm for the matching model.

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

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

4.4 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.

4.4.1 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 (Marti & Bunke, 2002) 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 (Grosicki *et al.*, 2009) 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°²àâçèéêëîïôùû`.
- The George Washington (GW) database (Rusinol *et al.*, 2011) 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 (Almázan *et al.*, 2014a).

- The Lord Byron (LB) database (Rusinol *et al.*, 2011) is a set of English typewritten texts containing 4988 word instances. This database has a vocabulary of 1569 words.
- The IFN/ENIT database (M. Pechwitz, 2002) 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.

4.4.2 Evaluation protocol

As in Almázan *et al.* (2014a), 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), which is a standard measure in retrieval systems. Once again, we followed the protocol proposed in Almázan *et al.* (2014a). 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.

4.4.3 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 (Kingma & Ba, 2014)

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 $\times$ height and 0.05 $\times$ 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.

4.4.4 Results

4.4.4.1 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 4.4 gives the mAP of these two models, using Euclidean distance retrieval or the MLP matching model of Section 4.3.4, as well as 6 different approaches for this task : 1) simple Fisher Vector (FV) encoding of word images (Almázan *et al.*, 2014a), 2) FVs jointly embedded with PHOC word representations and attribute SVMs used for matching (Almázan *et al.*, 2014a), 3) PHOCNet (Sudholt & Fink, 2016), 4) the AlexNet architecture pre-trained on the ImageNet database and then fine-tuned using IAM training examples (Sharma & K., 2015), 5) Deep feature embedding using pre-trained CNN Krishnan *et al.* (2016), and 6) PHOCNet with the Temporal Pyramid Pooling (TPP) layer Sudholt & Fink (2018). Since no test set is provided for the GW dataset, we used four-fold cross validation approach as in (Almázan *et al.*, 2014a).

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

Tableau 4.4 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 (Almázan *et al.*, 2014a)	15.66%	–
Attribute SVMs (Almázan *et al.*, 2014a)	55.73%	93.04%
Deep feature embedding Krishnan *et al.* (2016)	84.24%	94.41%
PHOCNet (Sudholt & Fink, 2016)	72.51%	92.64%
Fine-tuned CNN (Sharma & K., 2015)	46.53%	–
TPP-PHOCNet (BCA) Sudholt & Fink (2018)	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%

performances : the BoC encoding performs better when using Euclidean distance, while PHOC is best for the MLP matching model.

Tableau 4.5 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 (Almázan *et al.*, 2014a)	62.72%	86.01%	13.08%
Multi-scale features (Mhiri *et al.*, 2017)	66.1%	–	–
Ours (Euclidean dist. + BoC)	**78.60**%	**94.69**%	**28.13**%

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 4.5, 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., $\sim$1.4%). This indicates that the proposed representation captures information that does not depend on language. Our transferred representation also outperforms the approach of our previous work (Mhiri *et al.*, 2017), which learns a multi-scale representation in an unsupervised manner using the spherical k-means algorithm.

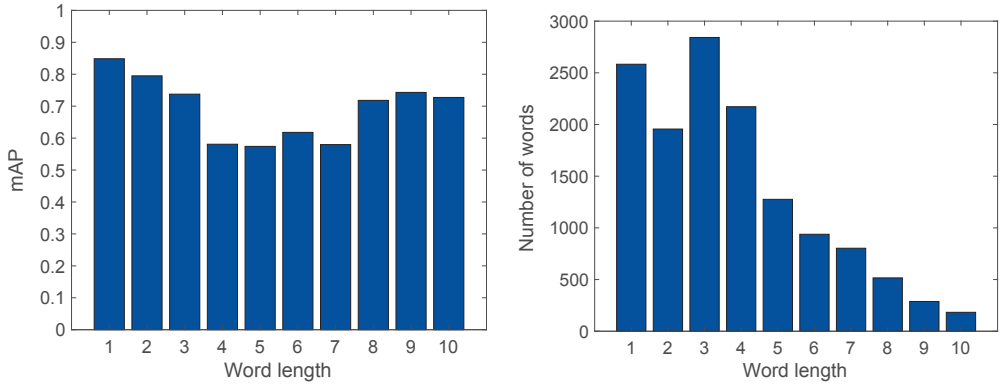


Figure 4.4 The mAP of the BoC-based model and occurrence frequency corresponding to different lengths of words in the IAM database.

Figure 4.4 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).

4.4.4.2 Query-by-string word spotting

Table 4.6 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 4.3.3). As in previous experiments, we compared our method against the joint embedding approach of Almazan et al. Almázan *et al.* (2014a), PHOCNet (Sudholt & Fink, 2016), Deep feature embedding using pre-trained CNN Krishnan *et al.* (2016), and PHOCNet with the Temporal Pyramid Pooling (TPP) layer Sudholt & Fink (2018).

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 (Sudholt & Fink, 2016) and of 16.95% over the approach in Almázan *et al.* (2014a). Likewise, the PHOC-based model with MLP matching gives an improvement of 8.91% over PHOCNet (Sudholt & Fink, 2016), 18.16% over the approach of Almázan *et al.* (2014a), and 0.3% over the method of Krishnan *et al.* (2016). 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.

Tableau 4.6 Query-by-string (QBS) word spotting performance in terms of mAP on the IAM database.

Method	**IAM test set**
Attribute SVMs (Almázan *et al.*, 2014a)	73.72%
PHOCNet (Sudholt & Fink, 2016)	82.97%
Deep feature embedding Krishnan *et al.* (2016)	91.58%
TPP-PHOCNet Sudholt & Fink (2018)	**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%

To illustrate the learned representation, Figure 4.5 (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. 4.5 (right).

For instance, the 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’.

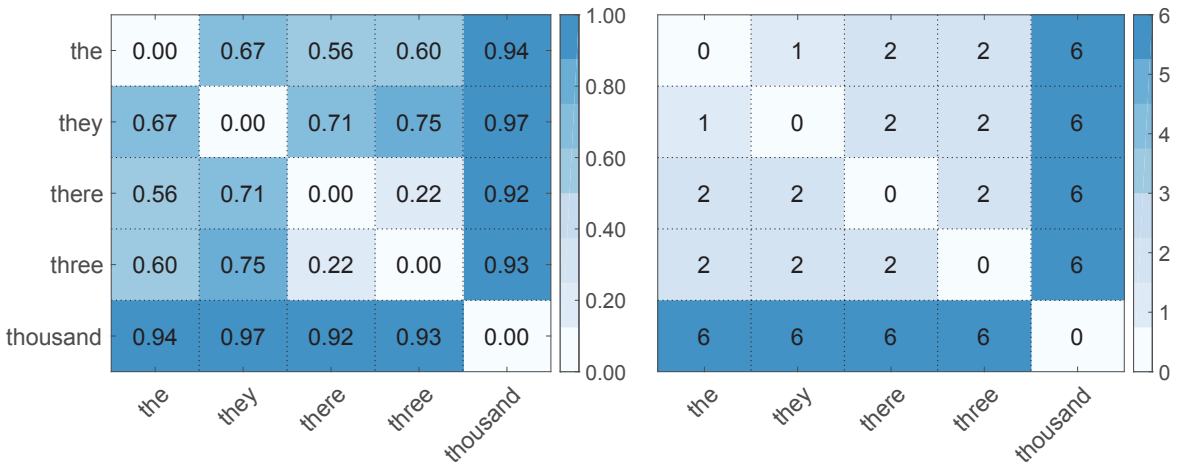


Figure 4.5 (**left**) L2 distance between the normalized representations of five words, obtained with the BoC-based model. (**right**) Levenshtein edit distance between same words. Words were selected to have various levels of spelling similarity.

Table 4.7 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.

Tableau 4.7 Query-by-string (QBS) word spotting performance in terms of mAP@k on the IAM database.

	mAP@5	**mAP@10**	**mAP@15**	**mAP@20**	**mAP@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%

4.4.4.3 Word recognition

Table 4.8 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 Almázan *et al.* (2014a), which also employs a joint embedding strategy, as well the HMM-based techniques described in Bianne-Bernard *et al.* (2011) and Espana-Boquera *et al.* (2011). We note that a direct comparison to Espana-Boquera *et al.* (2011) 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 (Bluche *et al.*, 2013b) or on Bidirectional LSTMs (Bluche *et al.*, 2014; Pham *et al.*, 2013). 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.

Tableau 4.8 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 (Bianne-Bernard *et al.*, 2011)	14.7%	–	21.9%	–
HMM (Espana-Boquera *et al.*, 2011)	16.8%	–	21.2%	9.1%
CNN-GHMM (Bluche *et al.*, 2013b)	**9.2**%	–	20.5%	–
BLSTM (Bluche *et al.*, 2014)	11.8%	3.7%	**11.9**%	**4.9**%
BLSTM (Pham *et al.*, 2013)	12.3%	**3.3**%	13.6%	5.1%
Attribute SVMs (Almázan *et al.*, 2014a)	–	–	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%

The word error rates and number of incorrect IAM word recognitions, for different word lengths, are shown in Fig. 4.6. 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 similar-looking punctuation characters (i.e., ‘.’ versus ‘,’ or ‘ :’ versus ‘ ;’).

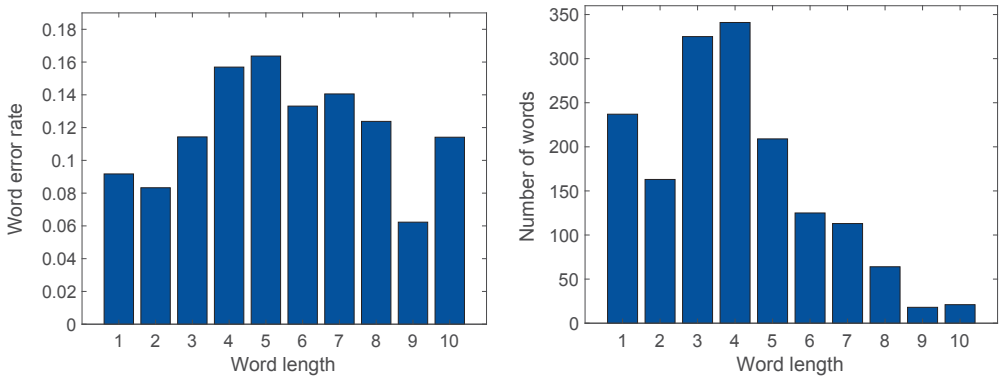


Figure 4.6 The word error rates and number of incorrect word recognitions obtained by the BoC-based model, for different lengths of words in the IAM database.

4.4.4.4 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.

4.5 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 Almázan *et al.* (2014a). 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 **?**, 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.

CHAPITRE 5

CONCLUSION GÉNÉRALE

Dans cette thèse, nous avons répondu aux questions de recherche relatives à la définition des représentations robustes pour les formes manuscrites. Les représentations proposées sont utilisées pour concevoir un système de reconnaissance et de repérage d'écriture manuscrite dans les documents numérisés.

Ce chapitre présente tout d'abord une résume des contributions faites dans le cadre de cette thèse. Par la suite, les limitations de ces contributions et des recommandations à suivre dans de futurs travaux sont discutées. À la fin, une liste complète des articles publiés dans le cadre de cette thèse est présentée.

5.1 Résumé des contributions

Dans une première contribution, une représentation non supervisée profonde est proposée pour la tâche de repérage de mots manuscrits. Cette représentation a l'avantage d'être définie d'une manière non supervisée, ce qui évite la nécessité d'avoir des données annotées pour l'entraînement. Aussi, elle se calcule rapidement et est de taille compacte, permettant ainsi de repérer des mots efficacement dans un scénario multi-auteurs et à grande échelle.

Dans une deuxième contribution, un modèle de bout en bout est développé pour la reconnaissance de mots manuscrits. Ce modèle est un réseau de neurones convolutifs qui prend en entrée l'image d'un mot et produit en sortie une représentation du texte reconnu. Cette représentation du texte capture la distribution des sous-séquences de caractères dans le corpus d'entraînement, et permet au modèle entraîné de transférer cette connaissance à de nouveaux mots contenant les mêmes sous-séquences. Ce modèle fournit des résultats compétitifs avec une grande efficacité de calcul.

Dans une troisième et dernière contribution, un modèle de bout en bout est proposé pour résoudre simultanément les tâches de repérage et de reconnaissance. Ce modèle intègre conjoin-

tement les textes et les images de mots dans un seul espace vectoriel grâce à une architecture qui combine les réseaux convolutifs et récurrents. Le modèle proposé est entraîné de manière à ce que l'image d'un mot et son texte soient projetés au même point. Dans l'espace vectoriel appris, les tâches de repérage et de reconnaissance peuvent être traitées efficacement comme un problème de recherche des plus proches voisins. Les résultats sur cinq bases de données ont montré la grande efficacité de ce modèle pour la tâches de repérage. De plus, ce modèle a produit des performances compétitives pour la tache de reconnaissance.

5.2 Limitations et recommandations

Les différents modèles de reconnaissance et de repérage proposés dans cette thèse traitent les images segmentées de mots. Cependant, la segmentation d'un document numérisé en un ensemble de mots est une tâche délicate puisque les mots ne sont pas toujours séparables et différents mots peuvent se chevaucher. Ainsi, une mauvaise segmentation peut entraîner des taux d'erreur élevés dans les traitements ultérieurs. Cette tâche pourrait être considérée dans d'autres projets de recherche car aucun modèle fiable de segmentation de mots n'est disponible à ce jour pour les documents manuscrits.

À travers cette thèse, différentes représentations pour les formes manuscrites ont été explorées. Ces représentations se basent principalement sur des modèles d'apprentissage profond composés d'une hiérarchie des couches de traitement non linéaire. Les paramètres de ces couches sont appris soit d'une façon supervisée en utilisant de données étiquetées (Chapitres 3 et 4), ou d'une façon non supervisée à l'aide de données non étiquetées (Chapitre 2). La représentation proposée au Chapitre 2 est une représentation non supervisée qui a démontré une grande robustesse à la variabilité intra-classe de formes manuscrites. Néanmoins, les performances de cette représentation sont encore limitées par rapport aux représentations définies de façon supervisée. Cela prouve l'importance de données étiquetées pour obtenir des résultats compétitifs. Une autre faiblesse de cette représentation non supervisée réside également dans l'algorithme de regroupement *spherical k-means*, qui utilise une initialisation aléatoire des groupes et peut converger vers un minimal local au lieu de la solution optimale.

La représentation proposée au Chapitre 4 est obtenue de manière supervisée via un réseau de neurones convolutifs entraîné à détecter les différentes formes de caractères. Cette représentation a démontré une grande efficacité dans les tâches de repérage et de reconnaissance de mots. Elle peut également être transférée à travers différents corpus de documents, surtout si ces corpus sont dans la même langue. Cette transférabilité pourrait être étudiée plus profondément dans des travaux futurs afin de définir des modèles de reconnaissance et repérage moins dépendants des données étiquetées.

Malgré leurs performances, les modèles de reconnaissance de mots proposés dans cette thèse partagent une limitation commune qui consiste à reconnaître uniquement un vocabulaire fermé lié à un dictionnaire donné. Il serait intéressant d'adapter ces modèles à la reconnaissance de mots hors vocabulaire. Cette reconnaissance hors vocabulaire pourrait également permettre la correction d'autres erreurs en utilisant des modèles spécifiques de langage.

5.3 Liste de Publications

Articles dans des revues avec comité de lecture

- **M. Mhiri**, S. Abuelwafa, C. Desrosiers, M. Cheriet, "Hierarchical representation learning using spherical k-means for segmentation-free word spotting", Pattern Recognition Letters, volume 101, 2018, pages 52-59.
- **M. Mhiri**, C. Desrosiers, M. Cheriet, "Convolutional pyramid of bidirectional character sequences for the recognition of handwritten words", Pattern Recognition Letters, volume 111, 2018, pages 87-93.
- **M. Mhiri**, C. Desrosiers, M. Cheriet, "Word spotting and recognition via a joint deep embedding of image and text", Pattern Recognition, (sous révision).

Communications dans des conférences internationales

- **M. Mhiri**, M. Cheriet, C. Desrosiers, “Query-by-example word spotting using multiscale features and classification in the space of representation differences”, IEEE International Conference on Image Processing (ICIP), 1112-1116, 2017.
- **M. Mhiri**, S. Abuelwafa, C. Desrosiers, M. Cheriet, “Footnote-based document image classification using 1D convolutional neural networks and histograms”, Image Processing Theory Tools and Applications (IPTA), 1-5, 2017.

BIBLIOGRAPHIE

Aggarwal, C. C., Hinneburg, A. & Keim, D. A. (2001). On the surprising behavior of distance metrics in high dimensional space. *Lecture notes in computer science*.

Almázan, J., Gordo, A., Fornés, A. & Valveny, E. (2014a). Word spotting and recognition with embedded attributes. *IEEE transactions on pattern analysis and machine intelligence*.

Almázan, J., Gordo, A., Fornés, A. & Valveny, E. (2014b). Segmentation-free word spotting with exemplar SVMs. *Pattern recognition*.

Alsharif, O. & Pineau, J. (2013). End-to-end text recognition with hybrid HMM maxout models. *Corr*.

Baoguang Shi,Xiang Bai, C. Y. (2015). An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *Corr*.

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and trends in machine learning*.

Bengio, Y., Courville, A. C. & Vincent, P. (2013). Representation learning : A review and new perspectives. *IEEE trans. pattern anal. mach. intell.*

Bianne-Bernard, A. L., Menasri, F., Mohamad, R. A.-H., Mokbel, C., Kermorvant, C. & Likforman-Sulem, L. (2011). Dynamic and contextual information in HMM modeling for handwritten word recognition. *IEEE transactions on pattern analysis and machine intelligence*.

Bianne-Bernard, F. M. . J. L. . A.-L. & Kermorvant, C. (2012). The A2iA french handwriting recognition system at the Rimes-ICDAR2011 competition. *Document recognition and retrieval xix*.

Bissacco, A., Cummins, M., Netzer, Y. & Neven, H. (2013). PhotoOCR : Reading text in uncontrolled conditions. *Proceedings of the ieee international conference on computer vision*, pp. 785–792.

Bluche, T., Ney, H. & Kermorvant, C. (2013a). Feature extraction with convolutional neural networks for handwritten word recognition. *International conference on document analysis and recognition*.

Bluche, T., Ney, H. & Kermorvant, C. (2013b). Tandem HMM with convolutional neural network for handwritten word recognition. *IEEE international conference on acoustics, speech and signal processing*.

Bluche, T., Ney, H. & Kermorvant, C. (2014). A comparison of sequence-trained deep neural networks and recurrent neural networks optical modeling for handwriting recognition. *International conference on statistical language and speech processing*.

Bluche, T. (2015). *Deep neural networks for large vocabulary handwritten text recognition*. (Thèse de doctorat).

Boureau, Y. L., Bach, F., LeCun, Y. & Ponce, J. (2010). Learning mid-level features for recognition. *Computer vision and pattern recognition*.

Candes, E. J., Eldar, Y. C., Strohmer, T. & Voroninski, V. (2015). Phase retrieval via matrix completion. *Siam review*, 57(2), 225–251.

Chherawala, Y., Roy, P. P. & Cheriet, M. (2013). Feature design for offline arabic handwriting recognition : Handcrafted vs automated. *International conference on document analysis and recognition*.

Chherawala, Y. & Cheriet, M. (2012). W-TSV : weighted topological signature vector for lexicon reduction in handwritten Arabic documents. *Pattern recognition*.

Chung, J., Gülçehre, Ç., Cho, K. & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *Corr*.

Coates, A. & Ng, A. Y. (2011). Selecting receptive fields in deep networks. *Advances in neural information processing systems*.

Coates, A., Carpenter, B., Case, C., Satheesh, S., Suresh, B., Wang, T., Wu, D. J. & Ng, A. Y. (2011a). Text detection and character recognition in scene images with unsupervised feature learning. *International conference on document analysis and recognition*.

Coates, A., Ng, A. Y. & Lee, H. (2011b). An analysis of single-layer networks in unsupervised feature learning. *Proceedings of the fourteenth international conference on artificial intelligence and statistics*.

Csurka, G., Dance, C. R., Fan, L., Willamowski, J. & Bray, C. (2004). Visual categorization with bags of keypoints. *In workshop on statistical learning in computer vision, ECCV*.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K. & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the american society for information science*.

Dolz, J., Desrosiers, C. & Ayed, I. B. (2017). 3D fully convolutional networks for subcortical segmentation in MRI : A large-scale study. *Neuroimage*.

El-Hajj, R., Mokbel, C. & Likforman-Sulem, L. (2008). Recognition of Arabic handwritten words using contextual character models. *Document recognition and retrieval*.

Espana-Boquera, S., Castro-Bleda, M. J., Gorbe-Moya, J. & Zamora-Martinez, F. (2011). Improving offline handwritten text recognition with hybrid HMM/ANN models. *IEEE transactions on pattern analysis and machine intelligence*.

Fink, G. & Plotz, T. (2007). On the use of context-dependent modeling units for hmm-based offline handwriting recognition. *International conference on document analysis and recognition*.

Fischer, A., Keller, A., Frinken, V. & Bunke, H. (2012). Lexicon-free handwritten word spotting using character hmms. *Pattern recognition letters*.

Fischer, A. (2012). *Handwriting recognition in historical documents*. (Thèse de doctorat).

Fischler, M. A. & Bolles, R. C. (1981). Random sample consensus : A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*.

Frinken, V., Fischer, A., Manmatha, R. & Bunke, H. (2012). A novel word spotting method based on recurrent neural networks. *IEEE transactions on pattern analysis and machine intelligence*.

Gatos, B. & Pratikakis, I. (2009). Segmentation-free word spotting in historical printed documents. *International conference on document analysis and recognition*.

Giotis, A. P., Sfikas, G., Gatos, B. & Nikou, C. (2017). A survey of document image word spotting techniques. *Pattern recognition*.

Goodfellow, I. J., Bulatov, Y., Ibarz, J., Arnoud, S. & Shet, V. (2013). Multi-digit number recognition from street view imagery using deep convolutional neural networks. *arxiv preprint arxiv :1312.6082*.

Graves, A., Liwicki, M., Fernandez, S., Bertolami, R., Bunke, H. & Schmidhuber, J. (2009a). A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence*.

Graves, A., Fernández, S. & Gomez, F. (2006). Connectionist temporal classification : Labelling unsegmented sequence data with recurrent neural networks. *International conference on machine learning*.

Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H. & Schmidhuber, J. (2009b). A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence*.

Grosicki, E., Carre, M., Brodin, J.-M. & Geoffrois, E. (2009). Results of the RIMES evaluation campaign for handwritten mail processing. *International conference on document analysis and recognition*.

Hadsell, R., Chopra, S. & LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, 2, 1735–1742.

He, K., Zhang, X., Ren, S. & Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *Ieee transactions on pattern analysis and machine intelligence*, 37, 1904-1916.

Hinton, G. E., Osindero, S. & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural comput.*

Hornik, K., Feinerer, I., Kober, M. & Buchta, C. (2012). Spherical k-means clustering. *Journal of statistical software*.

Howe, N. R. (2013). Document binarization with automatic parameter tuning. *International journal on document analysis and recognition (ijdar)*.

Ioffe, S. & Szegedy, C. (2015). Batch normalization : Accelerating deep network training by reducing internal covariate shift. *Corr*.

Jaderberg, M., Simonyan, K., Vedaldi, A. & Zisserman, A. (2014a). Synthetic data and artificial neural networks for natural scene text recognition. *Corr*.

Jaderberg, M., Vedaldi, A. & Zisserman, A. (2014b). Deep features for text spotting. *European conference on computer vision*.

Jaderberg, M., Simonyan, K., Vedaldi, A. & Zisserman, A. (2016). Reading text in the wild with convolutional neural networks. *International journal of computer vision*.

Jarrett, K., Kavukcuoglu, K., Ranzato, M. & LeCun, Y. (2009). What is the best multi-stage architecture for object recognition ? *International conference on computer vision*.

Jegou, H., Douze, M. & Schmid, C. (2011). Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*.

Jegou, H., Douze, M., Schmid, C. & Pérez, P. (2010). Aggregating local descriptors into a compact image representation. *Computer vision and pattern recognition*.

Kaiming He,Qifa Ke, J. S. (2013). Optimized product quantization for approximate nearest neighbor search. *IEEE computer society*.

Kingma, D. P. & Ba, J. (2014). Adam : A method for stochastic optimization. *Corr*.

Konidaris, T., Kesidis, A. L. & Gatos, B. (2016). A segmentation-free word spotting method for historical printed documents. *Pattern analysis and applications*.

Krishnan, P., Dutta, K. & Jawahar, C. V. (2016). Deep feature embedding for accurate recognition and retrieval of handwritten text. *International conference on frontiers in handwriting recognition (icfhr)*, 289-294.

Krishnan, P. & Jawahar, C. V. (2016). Matching handwritten document images. *Eccv 2016*, 766–782.

Krizhevsky, A. & Hinton, G. (2009). Learning multiple layers of features from tiny images. *Technical report, university of toronto*.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 1097-1105.

L. Gomez, M. R. & Karatzas, D. (2018). LSDE : Levenshtein space deep embedding for query-by-string word spotting. *017 14th iapr international conference on document analysis and recognition (icdar)*, 499-504.

lan Boureau, Y., Ponce, J. & Lecun, Y. (2010). A theoretical analysis of feature pooling in visual recognition. *International conference on machine learning*.

Lazebnik, S., Schmid, C. & Ponce, J. (2006). Beyond bags of features : Spatial pyramid matching for recognizing natural scene categories. *Computer vision and pattern recognition*.

Lecun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*.

Lee, H., Battle, A., Raina, R. & Ng, A. Y. (2006). Efficient sparse coding algorithms. *Advances in neural information processing systems*.

Lee, H., Ekanadham, C. & Ng, A. Y. (2008). Sparse deep belief net model for visual area v2. *Advances in neural information processing systems*.

Levenshtein, V. I. (2012). Binary codes capable of correcting deletions, insertions and reversals. *Soviet physics doklady*, 10, 707.

Li, F.-F. & Perona, P. (2005). A bayesian hierarchical model for learning natural scene categories. *Computer vision and pattern recognition*.

Likforman-Sulem, L., Zahour, A. & Taconet, B. (2007). Text line segmentation of historical documents : a survey. *International journal of document analysis and recognition*.

Louloudis, G., Gatos, B., Pratikakis, I. & Halatsis, C. (2009). Text line and word segmentation of handwritten documents. *Pattern recognition*.

M. Pechwitz,S. Snoussi Maddouri, V. M.-N. E. H. A. (2002). Ifn/enit-database of handwritten arabic words. *Colloque international francophone sur l'ecrit et le document*.

Malisiewicz, T., Gupta, A. & Efros, A. A. (2011). Ensemble of exemplar-svms for object detection and beyond. *International conference on computer vision, ICCV*.

Manmatha, R., Han, C. & Riseman, E. M. (1996). Word spotting : a new approach to indexing handwriting. *Computer vision and pattern recognition*.

Manmatha, R. & Rothfeder, J. L. (2005). A scale space approach for automatically segmenting words from historical handwritten documents. *Ieee transactions on pattern analysis and machine intelligence*, 27(8), 1212–1225.

Marti, U. & Bunke, H. (2002). The iam-database : An english sentence database for off-line handwriting recognition. *Int. journal on document analysis and recognition*.

Mhiri, M., Cheriet, M. & Desrosiers, C. (2017). Query-by-example word spotting using multiscale features and classification in the space of representation differences. 1112-1116.

Mhiri, M., Abuelwafa, S., Desrosiers, C. & Cheriet, M. (2018). Hierarchical representation learning using spherical k-means for segmentation-free word spotting. *Pattern recognition letters*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, pp. 3111–3119.

Mohamad, R. A.-H., Likforman-Sulem, L. & Mokbel, C. (2009). Combining slanted-frame classifiers for improved HMM-based Arabic handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence*.

Neumann, L. & Matas, J. (2013). Scene text localization and recognition with oriented stroke detection. *Proceedings of the ieee international conference on computer vision*, pp. 97–104.

P. Keaton, H. G. & Goodman, R. (1997). Keyword spotting for cursive document retrieval. *Document image analysis*.

Palacios, R., Gupta, A. & Wang, P. S. (2004). Handwritten bank check recognition of courtesy amounts. *International journal of image and graphics*.

Pedronette, D. C. G., Almeida, J. & da S. Torres, R. (2014). A scalable re-ranking method for content-based image retrieval. *Information sciences*.

Perronnin, F., Sánchez, J. & Mensink, T. (2010). Improving the fisher kernel for large-scale image classification. *European conference on computer vision*.

Pham, V., Kermorvant, C. & Louradour, J. (2013). Dropout improves recurrent neural networks for handwriting recognition. *Corr*.

Poznanski, A. & Wolf, L. (2016). CNN-N-gram for handwriting word recognition. *Conference on computer vision and pattern recognition*.

Pratikakis, I., Zagoris, K., Gatos, B., Puigcerver, J., Toselli, A. H. & Vidal, E. (2016). ICFHR 2016 handwritten keyword spotting competition (H-KWS 2016). *International conference on frontiers in handwriting recognition*.

Rath, M. T. & Manmatha, R. (2007). Word spotting for historical documents. *International journal of document analysis and recognition*.

Rodriguez-Serrano, J. A. & Perronnin, F. (2012). A model-based sequence similarity with application to handwritten word spotting. *IEEE transactions on pattern analysis and machine intelligence*.

Rothacker, L., Rusinol, M. & Fink, G. A. (2013). Bag-of-features HMMs for segmentation-free word spotting in handwritten documents.

Roy, P. P., Bhunia, A. K., Das, A., Dey, P. & Pal, U. (2016). Hmm-based indic handwritten word recognition using zone segmentation. *Pattern recognition*.

Rusinol, M., Aldavert, D., Toledo, R. & Llados, J. (2011). Browsing heterogeneous document collections by a segmentation-free word spotting method. *International conference on document analysis and recognition*.

Rusinol, M., Aldavert, D., Toledo, R. & Llados, J. (2015). Efficient segmentation-free keyword spotting in historical document collections. *Pattern recognition*.

Salton, G. & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing and management*.

Sauvola, J., Seppanen, T., Haapakoski, S. & Pietikainen, M. (1997). Adaptive document binarization. *International conference on document analysis and recognition*.

Scholkopf, B., Platt, J. & Hofmann, T. (2007). Efficient sparse coding algorithms. *Advances in neural information processing systems*.

Sharma, A. & K., P. S. (2015). Adapting off-the-shelf CNNs for word spotting & recognition. *Document analysis and recognition (icdar), 2015 13th international conference on*, 986–990.

Sharma, A. et al. (2015). Adapting off-the-shelf CNNs for word spotting & recognition. *Document analysis and recognition (icdar), 2015 13th international conference on*, pp. 986–990.

Simard, P. Y., Steinkraus, D. & Platt, J. (2003). Best practices for convolutional neural networks applied to visual document analysis. *Institute of electrical and electronics engineers, inc.*

Simonyan, K. & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *International conference on learning representations*.

Sivic, J. & Zisserman, A. (2003). Video Google : A text retrieval approach to object matching in videos.

Slimane, F., Zayene, O., Kanoun, S., Alimi, A. M., Hennebert, J. & Ingold, R. (2012). New features for complex arabic fonts in cascading recognition system. *International conference on pattern recognition*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014). Dropout : A simple way to prevent neural networks from overfitting. *Journal of machine learning research*.

Sudholt, S. & Fink, G. A. (2017). Evaluating word string embeddings and loss functions for CNN-based word spotting. *2017 14th iapr international conference on document analysis and recognition (icdar)*, 01, 493-498.

Sudholt, S. & Fink, G. A. (2016). PHOCNet : A deep convolutional neural network for word spotting in handwritten documents. *Corr*.

Sudholt, S. & Fink, G. A. (2018). Attribute CNNs for word spotting in handwritten documents. *International journal on document analysis and recognition (ijdar)*, 1433-2825.

Sun, Z., Jin, L., Xie, Z., Feng, Z. & Zhang, S. (2016). Convolutional multi-directional recurrent network for offline handwritten text recognition. *International conference on frontiers in handwriting recognition*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. (2015). Going deeper with convolutions. *IEEE conference on computer vision and pattern recognition (cvpr)*, 1-9.

van Gemert, J. C., Geusebroek, J. M., Veenman, C. J. & Smeulders, A. W. M. (2008). Kernel codebooks for scene categorization. *European conference on computer vision*.

Vincent, P., Larochelle, H., Bengio, Y. & Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. *International conference on machine learning*.

Vinciarelli, A. & Bengio, S. (2002). Offline cursive word recognition using continuous density hidden markov models trained with pca or ica features. *International conference on pattern recognition*.

Vinciarelli, A. & Luettin, J. (2001). A new normalization technique for cursive handwritten words. *Pattern recognition letters*.

Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on information theory*.

Wang, L., Li, Y. & Lazebnik, S. (2016). Learning deep structure-preserving image-text embeddings. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5005–5013.

Wilkinson, T., Lindström, J. & Brun, A. (2017). Neural Ctrl-F : Segmentation-free query-by-string word spotting in handwritten manuscript collections. *2017 ieee international conference on computer vision (iccv)*, 4443-4452.

Wu, Z., Ke, Q., Sun, J. & Shum, H.-Y. (2011). Scalable face image retrieval with identity-based quantization and multireference reranking. *IEEE transactions on pattern analysis and machine intelligence*.

Wöllmer, M., Schuller, B. & Rigoll, G. (2013). Keyword spotting exploiting long short-term memory. *Speech communication*.

Yan, F. & Mikolajczyk, K. (2015). Deep correlation for matching images and text. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3441–3450.

Youssouf, C. (2016). *Feature design and lexicon reduction for efficient off-line handwriting recognition*. (Thèse de doctorat).

Zeiler, M. D. (2012). ADADELTA : an adaptive learning rate method. *Corr*, abs/1212.5701.

Zhang, X. & Tan, C. L. (2013). Segmentation-free keyword spotting for handwritten documents based on heat kernel signature. *International conference on document analysis and recognition*.

Zhong, Z., Pan, W., Jin, L., Mouchère, H. & Viard-Gaudin, C. (2016). Spottingnet : Learning the similarity of word images with convolutional neural network for word spotting in handwritten historical documents. *Frontiers in handwriting recognition (icfhr), 2016 15th international conference on*, pp. 295–300.