Embedded Arabic text detection and recognition in videos

Sonia Yousfi

N°d'ordre NNT : 2016LYSEI069

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée au sein de

Orange Labs - France Télécom R&D, Rennes
et INSA de Lyon

Ecole Doctorale EDA 512
Informatique et Mathématiques de Lyon

Spécialité de doctorat :
Discipline : Informatique

Soutenue publiquement le 06/07/2016, par :
Sonia Yousfi

Embedded Arabic text detection and recognition in videos

Devant le jury composé de :

M. CANU, Stéphane	PRU, INSA Rouen	Rapporteur
M. WENDLING, Laurent	PRU, Université Paris Descartes	Rapporteur
M. THIRAN, Jean-Philippe	PRU, EPFL Lausanne	Examinateur
M. COUASNON, Bertrand	MC/HDR, INSA de Rennes	Examinateur
M. GARCIA, Christophe	PRU, INSA de Lyon	Directeur de thèse
M. BERRANI, Sid-Ahmed	Chef d'équipe/HDR, Orange Labs Rennes	Co-encadrant

Département FEDORA – INSA Lyon - Ecoles Doctorales – Quinquennal 2016-2020

SIGLE	ECOLE DOCTORALE	NOM ET COORDONNEES DU RESPONSABLE
CHIMIE	**CHIMIE DE LYON** **http://www.edchimie-lyon.fr** Sec : Renée EL MELHEM Bat Blaise Pascal 3e etage secretariat@edchimie-lyon.fr Insa : R. GOURDON	M. Stéphane DANIELE Institut de Recherches sur la Catalyse et l'Environnement de Lyon IRCELYON-UMR 5256 Équipe CDFA 2 avenue Albert Einstein 69626 Villeurbanne cedex directeur@edchimie-lyon.fr
E.E.A.	**ELECTRONIQUE, ELECTROTECHNIQUE, AUTOMATIQUE** **http://edeea.ec-lyon.fr** Sec : M.C. HAVGOUDOUKIAN Ecole-Doctorale.eea@ec-lyon.fr	M. Gérard SCORLETTI Ecole Centrale de Lyon 36 avenue Guy de Collongue 69134 ECULLY Tél : 04.72.18 60.97 Fax : 04 78 43 37 17 Gerard.scorletti@ec-lyon.fr
E2M2	**EVOLUTION, ECOSYSTEME, MICROBIOLOGIE, MODELISATION** **http://e2m2.universite-lyon.fr** Sec : Safia AIT CHALAL Bat Darwin - UCB Lyon 1 04.72.43.28.91 Insa : H. CHARLES Safia.ait-chalal@univ-lyon1.fr	Mme Gudrun BORNETTE CNRS UMR 5023 LEHNA Université Claude Bernard Lyon 1 Bât Forel 43 bd du 11 novembre 1918 69622 VILLEURBANNE Cédex Tél : 06.07.53.89.13 e2m2@ univ-lyon1.fr
EDISS	**INTERDISCIPLINAIRE SCIENCES-SANTE** **http://www.ediss-lyon.fr** Sec : Safia AIT CHALAL Hôpital Louis Pradel - Bron 04 72 68 49 09 Insa : M. LAGARDE Safia.ait-chalal@univ-lyon1.fr	Mme Emmanuelle CANET-SOULAS INSERM U1060, CarMeN lab, Univ. Lyon 1 Bâtiment IMBL 11 avenue Jean Capelle INSA de Lyon 696621 Villeurbanne Tél : 04.72.68.49.09 Fax :04 72 68 49 16 Emmanuelle.canet@univ-lyon1.fr
INFOMATHS	**INFORMATIQUE ET MATHEMATIQUES** **http://infomaths.univ-lyon1.fr** Sec :Renée EL MELHEM Bat Blaise Pascal 3e etage infomaths@univ-lyon1.fr	Mme Sylvie CALABRETTO LIRIS – INSA de Lyon Bat Blaise Pascal 7 avenue Jean Capelle 69622 VILLEURBANNE Cedex Tél : 04.72. 43. 80. 46 Fax 04 72 43 16 87 Sylvie.calabretto@insa-lyon.fr
Matériaux	**MATERIAUX DE LYON** **http://ed34.universite-lyon.fr** Sec : M. LABOUNE PM : 71.70 –Fax : 87.12 Bat. Saint Exupéry Ed.materiaux@insa-lyon.fr	M. Jean-Yves BUFFIERE INSA de Lyon MATEIS Bâtiment Saint Exupéry 7 avenue Jean Capelle 69621 VILLEURBANNE Cedex Tél : 04.72.43 71.70 Fax 04 72 43 85 28 Ed.materiaux@insa-lyon.fr
MEGA	**MECANIQUE, ENERGETIQUE, GENIE CIVIL, ACOUSTIQUE** **http://mega.universite-lyon.fr** Sec : M. LABOUNE PM : 71.70 –Fax : 87.12 Bat. Saint Exupéry mega@insa-lyon.fr	M. Philippe BOISSE INSA de Lyon Laboratoire LAMCOS Bâtiment Jacquard 25 bis avenue Jean Capelle 69621 VILLEURBANNE Cedex Tél : 04.72 .43.71.70 Fax : 04 72 43 72 37 Philippe.boisse@insa-lyon.fr
ScSo	**ScSo*** **http://recherche.univ-lyon2.fr/scso/** Sec : Viviane POLSINELLI Brigitte DUBOIS Insa : J.Y. TOUSSAINT viviane.polsinelli@univ-lyon2.fr	Mme Isabelle VON BUELTZINGLOEWEN Université Lyon 2 86 rue Pasteur 69365 LYON Cedex 07 Tél : 04.78.77.23.86 Fax : 04.37.28.04.48

*ScSo : Histoire, Géographie, Aménagement, Urbanisme, Archéologie, Science politique, Sociologie, Anthropologie

Embedded Arabic text detection and recognition in videos

Sonia YOUSFI

JOINT THESIS BETWEEN ORANGE LABS and INSA de LYON

To my parents

To my sister and brothers

To the one who is always by my side day and night...

Acknowledgements

I would like to express my special gratitude to my supervisors Christophe Garcia and Sid-Ahmed Berrani for their support and guidance. Thank you Christophe for all these years of learning and experience. Thank you my professor for your patience, your encouragement, for giving me this chance to work with you even before the thesis and to discover the field of machine learning through the eye of an expert like you. Sid-Ahmed, thank you for giving me the opportunity to pursue my research at Orange Labs. Thank you for your advices and constructive criticism. I learned a lot from your efficiency and organization in work as the head of the MAS team at Orange Labs.

I would also like to thank the jury members, Mr. Laurent Wendling, Mr. Stéphane Canu, Mr. Jean-Philippe Thiran and Mr. Bertrand Coüasnon for their precious time reading my thesis and for their constructive comments during my PhD defense.

I would to express my sincere thanks to Franck Mamalet for his excellent scientific collaboration, advices and guidance. I would also to thank all my colleagues at Orange Labs who, one way or another, have helped and supported me in this work, especially during the data collection stage. I would also like to thank my friends who have supported me during these years.

Finally, I would say thank you to my parents for their love, you are always there for me. My special thanks also go to my lovely brothers and to my sister Sara. Thank you my dear Sara for you wise counsel, sympathetic ear and for everything you have done for me.

Abstract

This thesis focuses on Arabic embedded text detection and recognition in videos. Different approaches robust to Arabic text variability (fonts, scales, sizes, etc.) as well as to environmental and acquisition condition challenges (contrasts, degradation, complex background, etc.) are proposed.

We introduce different machine learning-based solutions for robust text detection without relying on any pre-processing. The first method is based on Convolutional Neural Networks (ConvNet) while the others use a specific boosting cascade to select relevant hand-crafted text features.

For the text recognition, our methodology is segmentation-free. Text images are transformed into sequences of features using a multi-scale scanning scheme. Standing out from the dominant methodology of hand-crafted features, we propose to learn relevant text representations from data using different deep learning methods, namely Deep Auto-Encoders, ConvNets and unsupervised learning models. Each one leads to a specific OCR (Optical Character Recognition) solution. Sequence labeling is performed without any prior segmentation using a recurrent connectionist learning model. Proposed solutions are compared to other methods based on non-connectionist and hand-crafted features. In addition, we propose to enhance the recognition results using Recurrent Neural Network-based language models that are able to capture long-range linguistic dependencies. Both OCR and language model probabilities are incorporated in a joint decoding scheme where additional hyper-parameters are introduced to boost recognition results and reduce the response time.

Given the lack of public multimedia Arabic datasets, we propose novel annotated datasets issued from Arabic videos. The OCR dataset, called ALIF, is publicly available for research purposes. To the best of our knowledge, it is the first public dataset dedicated for Arabic video OCR. Our proposed solutions were extensively evaluated. Obtained results highlight the genericity and the efficiency of our approaches, reaching a word recognition rate of 88.63% on the ALIF dataset and outperforming well-known commercial OCR engine by more than 36%.

Résumé

Cette thèse s'intéresse à la détection et la reconnaissance du texte arabe incrusté dans les vidéos. Dans ce contexte, nous proposons différents prototypes de détection et d'OCR vidéo (Optical Character Recognition) qui sont robustes à la complexité du texte arabe (différentes échelles, tailles, polices, etc.) ainsi qu'aux différents défis liés à l'environnement vidéo et aux conditions d'acquisitions (variabilité du fond, luminosité, contraste, faible résolution, etc.).

Nous introduisons différents détecteurs de texte arabe qui se basent sur l'apprentissage artificiel sans aucun prétraitement. Les détecteurs se basent sur des Réseaux de Neurones à Convolution (ConvNet) ainsi que sur des schémas de boosting pour apprendre la sélection des caractéristiques textuelles manuellement conçus.

Quant à notre méthodologie d'OCR, elle se passe de la segmentation en traitant chaque image de texte en tant que séquence de caractéristiques grâce à un processus de scanning. Contrairement aux méthodes existantes qui se basent sur des caractéristiques manuellement conçues, nous proposons des représentations pertinentes apprises automatiquement à partir des données. Nous utilisons différents modèles d'apprentissage profond, regroupant des Auto-Encodeurs, des ConvNets et un modèle d'apprentissage non-supervisé, qui génèrent automatiquement ces caractéristiques. Chaque modèle résulte en un système d'OCR bien spécifique. Le processus de reconnaissance se base sur une approche connexionniste récurrente pour l'apprentissage de l'étiquetage des séquences de caractéristiques sans aucune segmentation préalable. Nos modèles d'OCR proposés sont comparés à d'autres modèles qui se basent sur des caractéristiques manuellement conçues. Nous proposons, en outre, d'intégrer des modèles de langage (LM) arabes afin d'améliorer les résultats de reconnaissance. Nous introduisons différents LMs à base des Réseaux de Neurones Récurrents capables d'apprendre des longues interdépendances linguistiques. Nous proposons un schéma de décodage conjoint qui intègre les inférences du LM en parallèle avec celles de l'OCR tout en introduisant un ensemble d'hyper-paramètres afin d'améliorer la reconnaissance et réduire le temps de réponse.

Afin de surpasser le manque de corpus textuels arabes issus des contenus multimédia, nous mettons au point de nouveaux corpus manuellement annotés à partir des flux TV arabes. Le corpus conçu pour l'OCR, nommé ALIF et composé de 6,532 images de texte annotées, a été publié a des fins de recherche. Nos systèmes ont été développés et évalués sur ces corpus. L'étude des résultats a permis de valider nos approches et de montrer leurs efficacité et généricité avec plus de 97% en taux de détection, 88.63% en taux de reconnaissance mots sur le corpus ALIF dépassant ainsi un des systèmes d'OCR commerciaux les mieux connus par 36 points.

Contents

List of Figures

List of Tables

French summary

0.1 Introduction

Les vidéothèques numériques connaissent de nos jours une croissance énorme avec les outils de partage des vidéos, la délinéarisation des programmes TV et les systèmes de vidéos à la demande. Représenter, chercher et récupérer un tel type de données est devenu assez important et demande l'exploitation de toute information disponible à l'intérieur ou à l'extérieur de la vidéo. Le texte incrusté, artificiellement ajoutée à une vidéo, comme les noms des lieux et les sous-titres des journaux télévisés, est l'un des sémantiques de haut niveau les plus utilisées dans la structuration et la récupération des vidéos. Plusieurs travaux ont été proposés pour la détection et la reconnaissance des indices textuelles dans les contenus multimédia. Néanmoins, la plupart des systèmes, généralement dénommés des systèmes d'OCR (Optical Character Recognition), ont été dédiés pour le texte Latin bien que beaucoup d'autres langues représentent une grande partie des contenus multimédia telle que la langue arabe.

Dans cette thèse, nous nous sommes intéressés à la détection et à la reconnaissance du texte arabe incrusté dans les vidéos. L'accent est mis sur le texte arabe en particulier pour plusieurs raisons. Cette langue est utilisée par plus de la moitié d'un milliard de personnes dans le monde et les deux dernières décennies ont connu l'apparition de plusieurs chaînes arabes très grandes et populaires. En plus, le texte arabe présente plusieurs spécificités qui rendent sa détection et sa reconnaissance, en deuxième lieu, des tâches très difficiles et spécifiques nécessitant une étude à part (texte cursif, des formes plus variées que les textes Latin et Chinois, etc.). Les spécificités des telles masses de données nécessite bien évidemment des systèmes d'OCR sophistiqués qui peuvent alimenter automatiquement les outils de gestion de vidéo arabe par des métadonnées et booster ainsi le marché de services et d'applications dans cette optique.

Traiter la détection et la reconnaissance de texte dans les contenus vidéo n'est une tâche assez triviale. Outre les spécificités du texte arabe (telles que l'aspect cursif, la grande variation morphologique des lettres arabes qui peuvent changer de formes même selon la position de la lettre dans un mot, la variation des fontes styles, polices, etc.), s'ajoutent d'autres complexités liées à l'environnement vidéo (telles que la variation du fond, la luminosité non-uniforme, le contraste, les occlusions, etc.) et aux conditions d'acquisition (faible résolution, dégradations, bruit, etc.). Plus particulièrement pour l'OCR vidéo arabe, un problème majeur auquel nous nous sommes confrontés est le manque de corpus. En effet, pour l'OCR arabe, les corpus publiquement disponibles sont limités au texte manuscrit ou imprimé dans les documents scannés tels que les livres, les magazines et les documents administratifs. Au début de ce projet, aucun

corpus n'a été disponible pour la détection ou la reconnaissance du texte arabe dans les contenus multimédia y compris les photos et les vidéos.

Dans ce travail, nous traitons presque la chaîne complète de l'OCR vidéo depuis la détection jusqu'à la reconnaissance de texte. Notre démarche commence par une étape primordiale de construction de corpus de textes annotés issu des flux TV arabes afin de mettre en place et évaluer nos systèmes. À l'issu de cette étape, nous proposons différents corpus: un pour la détection et l'autre pour la reconnaissance ainsi qu'un autre corpus de caractères arabes. À la base de ces données annotées, nous mettons en place des systèmes complets de détection et de reconnaissance de texte arabe incrusté dans les vidéos qui sont robustes aux différentes complexités déjà citées. Pour la phase de détection, nous proposons trois solutions à base d'apprentissage artificiel pour la classification des régions de l'image. Un scanning optimisé multi-échelle est ensuite appliqué suivi d'un algorithme de regroupement de votes pour une localisation précise du texte. L'apprentissage de classification est basé sur des techniques les mieux classées dans l'état de l'art. Nos techniques ne se basent sur aucun prétraitement des images en entrées. Les systèmes traitent directement des données brutes sans aucune à priori et font face aux différentes complexités du texte et de l'environnement vidéo grâce à une procédure d'entraînement qui se focalise sur le rejet des fausses alarmes les plus critiques (texte non-arabe, régions multi-lignes, textures semblables au texte arabe, etc.).

Concernant la phase de reconnaissance, notre approche globale proposée se passe de la segmentation en traitant la zone de texte comme une séquence de caractéristiques visuelles. Le problème de reconnaissance est ainsi transformé en une tâche de classification temporelle que nous traitons avec apprentissage connexionniste récurrent. Dans ce travail, nous nous passons aussi des représentations manuellement conçues des images de texte. Nous proposons plutôt des représentations avec des caractéristiques apprises de façon automatique. Nous utilisons pour cela différents modèles connexionniste neuronaux profonds qui apprennent l'extraction des modalités les plus pertinentes du texte de façon hiérarchique en se basant seulement sur des exemples. Chaque modèle donne une représentation bien spécifique des zones textuelles et ainsi, un système d'OCR différent. Afin de booster encore les résultats de reconnaissance obtenus, nous proposons une intégration de l'information linguistique en parallèle avec celle de l'OCR. Contrairement aux méthodes existantes qui se basent sur des modèles de langage fréquentiels, nous introduisons d'autres modèles neuronaux pouvant traiter les longues interdépendances, au niveau caractères, dans le langage. Nous proposons en outre un schéma de décodage conjoint des inférences linguistiques apprises et celle de la reconnaissance optique avec intégration de plusieurs hyper-paramètres afin de booster au plus les taux de reconnaissance tout en gardant un temps de réponse optimal. Nous menons des études expérimentales et comparatives exhaustives au niveau de chacune de ces étapes de détection et de reconnaissance tout en soulignant les avantages et les inconvénients des méthodes proposées. Dans ce chapitre, nous présentons une vue globale des systèmes proposés reflétant l'organisation du manuscrit de cette thèse. Dans la Section 0.2, nous décrivons brièvement les différents corpus construits et utilisés au cours de cette thèse. Ensuite, nous présentons dans la Section 0.3 les différentes approches de détection de texte utilisés. Les Sections 0.4 et 0.5 décrivent les différents systèmes de reconnaissance de texte arabe que nous proposons dans cette thèse et les méthodologies de leur amélioration en utilisant les modèles de langage. Nous donnons ensuite, dans la Section 0.6, une brève description de nos plans expérimentaux et un résumé des résultats obtenus. La Section 0.7 conclut ce travail avec quelques perspectives.

0.2 Les corpus

Comme nous l'avons précisé dans l'introduction, un des problèmes majeurs confrontés tout au long de ce travail est le manque de données i.e. corpus d'images de texte arabe issus des contenus multimédia (images et vidéos). En fait, les corpus arabes existants sont limités au texte manuscrit ou imprimé dans des documents scannés tels que les livres, les magazines, les documents administratifs, etc. Ce type de documents présente un environnement tout à fait différent de celui des vidéos et même des images de scènes naturelles. Pour mettre en place nos systèmes de détection et de reconnaissance, une première étape primordiale est de construire nos propres corpus issus des vidéos. Nous collectons des vidéos principalement à partir des flux TV arabe et nous préparons un logiciel d'annotation à cet effet. Un premier corpus est dédié pour la détection. Nous collectons pour cela des frames vidéo issues d'un ensemble de chaînes TV arabes et nous annotons les régions de texte arabe figurant dans ces vidéos. L'annotation est faite de façon à préciser pour chaque zone de texte sa position exacte dans l'image. Une partie des zones annotées est utilisée pour construire un ensemble d'apprentissage composé de 30,000 zones de texte pour la classification texte/non-texte. Les exemples négatifs (régions non-texte ou texte non-arabe) sont construits à partir d'un ensemble d'images de scène ne contenant pas du texte arabe. Une autre partie d'images annotées est utilisée pour le test des méthodes de détection proposées. Pour la reconnaissance, nous proposons un premier corpus de caractères arabes, nommé *ArabCharSet*, issus de 16 heures de flux TV arabe et de quelques images collectées du web. Nous annotons manuellement le texte arabe figurant sur les images et les frames de façon à spécifier les frontières des caractères. Un ensemble de 20,571 images de caractères arabes est extrait initialement à partir de ces images auxquelles nous appliquons quelques opérations de faible changement d'échelle, d'addition de bruit et d'inversion de couleurs pour atteindre en total 46,689 images de caractères arabes. Vu que les lettres arabes peuvent changer de forme selon leur position dans le texte, nous avons construit ce corpus de façon à prendre en compte toutes les morphologies des lettres qui sont représentées presque de manière équitable dans les corpus. Le corpus inclut aussi des images de caractères spéciaux comme les ponctuations et des images de chiffres arabes ainsi que des images d'une classe qui ne correspond à aucun caractère appelée classe '*Rubish*'. Ce corpus est dédié principalement pour la classification des caractères et la génération des caractéristiques textuelles. Un deuxième corpus de reconnaissance est construit à partir d'un ensemble de 64 vidéos issus de plusieurs chaînes TV arabes (Al Jazeera, Al Arabiya, France 24 Arabic, BBC Arabic and Al Hiwar) et quelques images du web. Le corpus, nommé *ALIF*, comprend 6,532 images de texte extraites de ces contenus et annotées en termes de transcription du texte qui y figure. Une partie du corpus comprend une annotation fine déterminant la position et la transcription de chaque caractère, syllabe et mot dans le texte. Ce sous-ensemble peut être utilisé pour développer des méthodes de reconnaissance basées sur la segmentation caractères. Afin de booster l'OCR arabe pour les contenus multimédias, ce corpus a été publié et mis à disposition pour les prochains travaux de recherche en OCR à des fins non-commerciales. Nous proposons aussi une organisation du corpus en des ensembles d'apprentissage et de test pour permettre des futures analyses comparatives des méthodes. Ce corpus comprend une grande variété de texte arabe et représente les défis majeurs qui peuvent être confrontés lors de l'OCR vidéo à savoir la complexité du texte (différentes fontes, styles, polices, etc.), des défis liés à l'environnement (la variation et la complexité du fond, la luminosité non-uniforme, la variation du contraste, des occlusions partielles du texte, etc.) ainsi que la complexité

au niveau des conditions d'acquisition (faible résolution, dégradation, bruit, etc.). La figure 1 illustre quelques exemples de ce corpus et la complexité qu'il présente.

FIGURE 1: Exemples d'images de texte issues du corpus ALIF.

0.3 La détection du texte arabe

Une grande partie des méthodes de détection de texte existantes sont basées sur des prétraitements et des règles géométriques. Ceci peut être une solution triviale pour des contenus multimédia simples avec un fond relativement uniforme et généralement où la binarisation donne des meilleurs résultats. Néanmoins, dans notre contexte, avec les défis liés aux contenus vidéo et aux propriétés du texte arabe, de telles méthodes ne sont pas assez génériques et efficaces. Pour cela, nous proposons des approches de détection qui se basent principalement sur l'apprentissage artificiel. Ceci permettra de définir de façon automatique l'ensemble des heuristiques discriminantes entre les régions de texte et le fond grâce à un processus d'apprentissage automatique à partir des exemples. Les modèles de détection comportent ainsi deux tâches principales: (1) la classification texte/non-texte des régions dans une frame vidéo et (2) la localisation précise des lignes de texte avec des boîtes englobantes, par exemple.

0.3.1 Les approches de classification

Pour la classification, nous proposons deux types d'approches: une approche connexionniste se basant sur les ConvNets et deux autres à base de Boosting. L'idée en général est d'apprendre des modèles de classification texte/non-texte à partir d'exemples. Comme nous avons déjà expliqué, nous nous intéressons, dans ce travail, à la détection du texte arabe plus spécifiquement. Les classifieurs sont ainsi conçus de manière à mieux discriminer entre les zones de texte arabe et les autres zones y compris le fond et le texte non-arabe.

0.3.1.1 L'approche basée Réseaux de Neurones à Convolution

Les ConvNets sont des modèles de classification neuronaux très puissants qui ont prouvé des meilleures performances dans plusieurs tâches de classification de formes (visages, caractères, objets, etc.). Leur structure leurs a permis d'être parmi les systèmes les plus robustes en termes de translation, changement d'échelle et distorsions et ceci grâce à trois concepts qu'ils disposent: l'utilisation de champs réceptifs locaux permettant de

détecter des caractéristiques locales, les poids partagés qui permettent d'appliquer la même recherche de ces caractéristiques sur toutes les positions dans une carte d'entrée, le sous-échantillonnage spatial réduisant la sensibilité aux variations de faible amplitude en translation, rotation et échelle et aux faibles distorsions. Grâce à une architecture qui intègre une organisation en cartes de caractéristiques (*feature maps*) et un processus d'apprentissage global qui vise à minimiser une seule fonction objectif, le réseaux est capable d'apprendre à la fois l'extraction des caractéristiques visuelles et la classification. Dans ce travail, nous proposons une première méthode de classification basée sur les ConvNets. Le réseau est entrainé en utilisant un ensemble d'apprentissage sous formes de patches de texte arabe (exemples positifs) et de patches de non-texte et texte non-arabe (exemples négatifs). L'apprentissage est supervisé et se base sur l'algorithme de rétropropagation du gradient en visant une sortie de 1 pour les exemples positifs et -1 pour les exemples négatifs. Au cours de l'apprentissage, le réseau explore et entraîne automatiquement ses propres extracteurs de caractéristiques de façon à minimiser la fonction de coût de classification (MSE) sans aucune *a priori* sur les caractéristiques et aucun prétraitement des données en entrée. Le ConvNet utilisé est composé de six couches et reçoit en entrée, au niveau de sa rétine de 32×64 pixels, des images ou patches d'apprentissage étiquetées. Les premières quatre couches représentent une alternance de couches de convolution et de sous-échantillonnage effectuant l'extraction des caractéristiques. Les deux dernières couches forment un simple perceptron multicouches (MLP) pour la classification. Plusieurs architectures ont été entraînées et évaluées en variant le nombre de cartes de convolution et de sous-échantillonnage à travers les couches. Les meilleurs résultats de classifications ont été obtenus avec un modèle qui comprend 5 cartes dans chacune des deux premières couches de convolution et de sous-échantillonnage, 2O cartes dans chacune des deux couches suivantes (de convolution et sous- échantillonnage) et 20 neurones dans la première couche de classification. La sortie du réseau avec une activation sigmoïde présente la classe prédite par le réseau (nous visons une valeur de 1 pour la classe texte arabe et -1 sinon).

Le modèle est entrainé avec 30,000 exemples de texte arabe comprenant une large variété de fontes, styles, échelles, etc. Au début de l'apprentissage, l'ensemble des exemples négatifs est alimenté avec 20,000 patches de non-texte représentant des zones de fond, de texte non-arabe, des zones multi-lignes, des zones partielles de textes, etc. Ce choix d'exemples négatifs est permet de booster la capacité du réseau à rejeter ces régions d'images ce qui améliore la précision de localisation. Pour améliorer encore la capacité de rejet du réseau, nous intégrons un mécanisme de bootstrap au cours de l'apprentissage. Cela consiste à alimenter l'ensemble des exemples d'apprentissage négatifs avec des fausses alarmes sélectionnées par le réseau à partir d'un ensemble d'image de scène ne contenant pas du texte. Cette phase de bootstrap est effectuée chaque 50 itérations d'entraînement et sert à focaliser l'apprentissage sur des exemples de plus en plus difficiles à classifier.

0.3.1.2 L'approche basée Boosting

Une deuxième catégorie de classifieurs que nous proposons est basée sur le Boosting. Cet algorithme est classé parmi les meilleures techniques de classification surtout en détection de visages avec les travaux de Viola Jones [VJ01a]. Contrairement aux ConvNets, le Boosting nécessite des caractéristiques manuellement conçues. Dans ce travail, nous proposons d'utiliser les caractéristiques *Multi-Block Local Binary Pattern* (MBLBP) basées sur un encodage de régions rectangulaires de plusieurs tailles de l'image par

l'opérateur Local Binary Pattern (LBP). Le but de l'application de Boosting est de sélectionner les caractéristiques les plus pertinentes pour une meilleure classification entre les zones de texte et de non-texte. Pour se faire, nous somme basés sur une architecture particulière de Boosting en cascade appelée le Boosting asymétrique multi-sorties (Multi-exit Asymmetric Boosting) [PHC08]. Dans cette structure en cascade, des classifieurs intermédiaires relativement forts sont représentés par un ensemble de nœuds. À chaque nœud, un algorithme de Boosting est appliqué pour sélectionner des caractéristiques considérées comme des classifieurs faibles. Au niveau de la cascade, nous utilisons l'algorithme de Gentleboost pour apprendre de façon hiérarchique ce processus de sélection des caractéristiques MBLBP les plus pertinentes menant à une meilleure classification texte/non-texte. Nous proposons de comparer ce classifieur avec un autre, plus connu, basé sur les caractéristiques de Haar et Adaboost avec la même structure en cascade.

Pendant l'entraînement, le modèle prend en entrée des images de texte et de non-texte présentées sous formes de caractéristiques MBLBP ou Haar. À chaque nœud, un algorithme de Boosting (Gentleboost ou Adaboost) s'occupe de la sélection des caractéristiques significatives et l'entraînement d'un nouveau classifieur plus au moins fort qui rejette les exemples négatifs et laisse passer, le maximum possible, des exemples de texte arabe au prochain nœud. La propriété la plus importante de l'architecture asymétrique multi-sortie est que chaque nœud calcule le score (la valeur de la confidence) à partir du premier classifieur faible. Par conséquence, étant donné le même nombre d'hypothèses faibles, chacun de ces nœuds peut utiliser plus de classifieurs faibles que dans la cascade de Viola Jones ce qui renforce en plus les classifieurs intermédiaires. Cette architecture permet aussi de garder toutes les informations précédentes dans le fil de l'entraînement ce qui offre plus de stabilité. Le nombre de classifieurs faibles à chaque nœud est déterminé selon un taux de détection et un taux de fausses alarmes ciblés. Une autre particularité de la détection de texte traitée par cette structure de classification est l'asymétrie. La première asymétrie dans ce problème de détection vient de la distribution inégale des données. Parmi les millions d'exemples générés à partir d'une image d'entrée, très peu d'entre elles contiennent du texte et du texte arabe en particulier. L'occurrence d'une zone de texte dans une image est un événement rare. La seconde asymétrie est liée à la différence entre les taux de détection et de rejet ciblés au cours de l'entraînement. D'une part, un taux de détection très haut est désiré et, d'autre part, un taux très faible de fausses alarmes est nécessaire pour une détection fiable.

L'apprentissage de ces classifieurs doit être fait tout en respectant toutes ces asymétries. Nous suivons pour cela la même stratégie des méthodes conventionnelles en fixant les taux de détection et de fausses alarmes ciblés : un rejet se fait seulement quand le taux de fausses alarmes est inférieur à une constante α_0 et le taux de détection est supérieur à $1 - \beta_0$, où α_0 et β_0 sont définies *a priori*. Pareil que l'apprentissage des ConvNets, nous utilisons le bootstrap aussi pour améliorer le taux de rejet des fausses alarmes. Le processus est appliqué à chaque nœud de la cascade.

0.3.2 La localisation du texte arabe

Une fois que les modèles de classification sont appris, la seconde phase consiste à les appliquer sur les frames vidéo afin de déterminer les zones de texte arabe les plus probables. Pour se faire nous utilisons un processus de scanning multi-échelle de chaque image afin que le classifieur puisse détecter les zones textuelles à différentes échelles et tailles.

Pour les méthodes à base de Boosting nous utilisons la technique des fenêtres glissantes ou le classifieur est appliqué à plusieurs positions de l'image. Vu que cette technique est très coûteuse en temps de réponse, nous utilisons la technique de l'Image Intégrale pour le calcul des caractéristiques sur toute l'image sans avoir à répéter les mêmes opérations à chaque position de la fenêtre. Pour les ConNets, cette procédure de scanning reste très coûteuse surtout que l'extraction des caractéristiques et la classification se font au sein du même modèle neuronal. Pour cela nous optons pour la solution de carte spatiale qui consiste à appliquer les filtres appris directement sur la totalité de l'image. Cela produit pour chaque image une carte 4 fois plus petite présentant les réponses du réseau (positions avec fortes probabilités d'existence du texte arabe). Cette procédure de scanning avec les classifieurs est appliquée à différentes échelles de l'image originale ce qui donne plusieurs cartes de votes. Ces réponses sont ensuite projetées sur une même carte correspondant à l'échelle originale et les réponses sont regroupées selon leurs proximités en espace et échelle via un algorithme *k means-like*. En s'appuyant sur un seuillage sur les scores des clusters obtenus, nous filtrons les fausses alarmes pour garder enfin les lignes de texte les plus probables.

0.4 La reconnaissance du texte arabe

Une fois détectées, les lignes de texte sont extraites pour passer à la deuxième étape de reconnaissance. Dans ce travail, notre méthodologie d'OCR se passe de tout prétraitement des zones de textes détectées en prenant directement les pixels des images comme entrées. Cette méthodologie se passe aussi de toute étape de segmentation en traitant l'image de texte comme une séquence de caractéristiques visuelles. Le schéma d'OCR proposé est illustré de façon plus claire par la Figure 2. Il est composé de deux étapes principales: (1) l'extraction des caractéristiques visuelles et (2) la classification temporelle ou l'étiquetage de séquence en vue de reconnaissance. La première étape vise à transformer l'image de texte en une séquence de caractéristiques les plus pertinentes possibles en termes de représentativité des différentes modalités du texte tout en conservant numériquement sa structure en caractères, syllabes, mots, etc. et sa différence par rapport au fond. La deuxième étape consiste à étiqueter séquentiellement les suites de caractéristiques pour chaque image de texte. Pour cette étape, notre méthode ne se base pas sur une segmentation préalable mais plutôt sur une classification temporelle des séquences de caractéristique sans à priori sur les frontières de chaque caractère. Cette classification se base sur un type particulier de Réseaux de Neurones Récurrents (RNN): les réseaux de neurones bidirectionnel à longue mémoire à court-terme (BLSTM) qui apprend l'étiquetage de séquence en utilisant une fonction objectif à base d'alignement entre les séquences de caractéristiques et les transcriptions de textes visées.

0.4.1 L'extraction des caractéristiques

Afin de présenter l'image de texte sous forme séquentielle, nous procédons à un scanning multi-échelle (cf. Figure 3) avec la technique des fenêtres glissantes. Nous utilisons différentes fenêtres à différents aspect-ratios pour balayer horizontalement chaque image de texte. Ceci aide à modéliser le texte à différentes échelles et positions. À chaque position du scanning, nous appliquons l'extraction de caractéristiques pour chaque fenêtre.

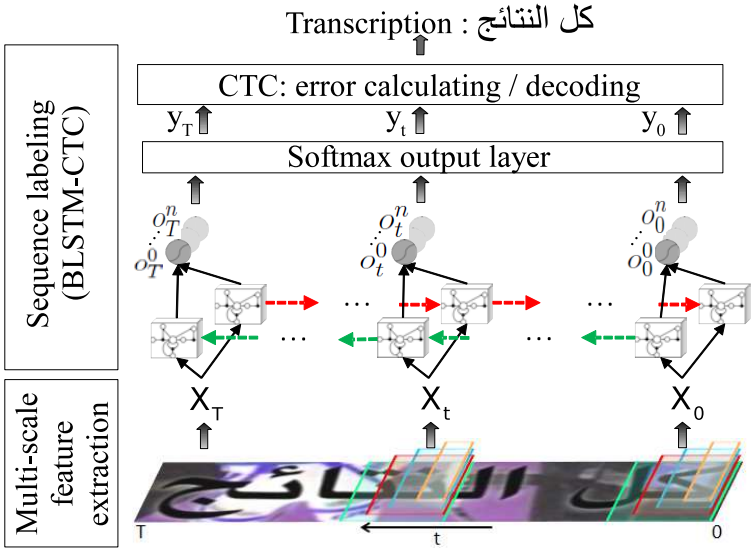


FIGURE 2: Schéma général de reconnaissance proposé.

Les vecteurs de caractéristiques ou descripteurs en sortie sont concaténés à chaque position (ou pas de scanning). Dans la phase d'extraction des caractéristiques, nous cherchons dans l'image de texte des informations saillantes qui peuvent servir pour la reconnaissance. Ces caractéristiques ou descripteurs doivent être robustes aux différentes complexités liées au texte et à l'environnement vidéo tout en gardant une représentativité des différents composants du texte (caractères, syllabes, mots, etc.) pour faciliter la classification temporelle dans une deuxième étape. Une façon pour se faire est d'utiliser des caractéristiques conçues manuellement. Cependant, ces caractéristiques varient d'un domaine à un autre. Elles dépendent de ce que l'être humain ou plutôt l'expert voit ce qui est important dans l'image. En plus, elles sont généralement adaptées au domaine ce qui réduit, de façon considérable, leur généricité.

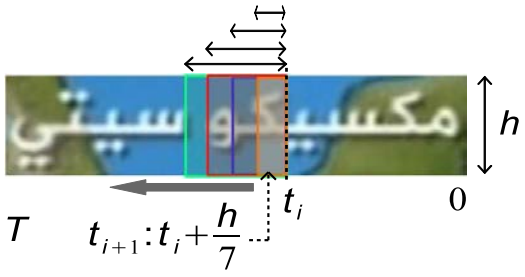


FIGURE 3: Procédure de scanning.

Pour remédier à ces inconvénients, nous proposons d'utiliser des descripteurs appris au lieu de les concevoir manuellement. En parle ainsi de '*feature engineering*' automatisé. En se basant sur des modèles d'apprentissage artificiel, nous proposons d'apprendre l'extraction des caractéristiques directement à partir d'exemples. Une fois appris, ces modèles sont appliqués au niveau des différentes fenêtres glissantes pour générer les

séquences de caractéristiques pour les différentes images de texte à des finalités de reconnaissance. Pour apprendre l'extraction de ces caractéristiques, nous nous somme basés principalement sur des modèles connexionnistes d'apprentissage artificiel (*deep learning*). Nous utilisons différents types de réseaux de neurones à architectures profondes faisant appel à différents schémas d'entraînement. L'apprentissage est basé sur des images de caractères arabes intégrant aussi des images de caractères spéciaux tels que la ponctuation et les chiffres arabes. Nous visons ainsi à explorer les différentes modalités de ces données qui représentent les composants élémentaires du texte et de les projeter dans un espace à faible dimension tout en gardant une certaine discrimination entre les caractères et une robustesse aux différentes variations des propriétés du texte, du fond et des conditions d'acquisition.

Nous utilisons, en premier lieu, des modèles d'auto-encodage à base de *Deep Belief Networks* [Hin07] et de perceptrons multicouches (MLP). Chacun de ces modèles prend en entrée les images de caractères arabes et vise à les reconstruire. L'architecture est composée d'un encodeur qui est entraîné à projeter une image en entrée dans un espace de caractéristiques, et d'un décodeur qui reconstruit cette entrée à partir de ses coordonnées dans cet espace. Cette représentation intermédiaire dans l'espace de projection est généralement appelée '*code*'. C'est ce code qui est retenu comme caractéristiques une fois le modèle est appris. Pour le DBN, l'apprentissage consiste en une première phase de pré-entraînement non-supervisé avec une pile de machines de Boltzman restraintes (RMB) pour une bonne initialisation des poids du modèle et d'une deuxième phase de redressement de ces poids avec un apprentissage qui vise la reconstruction des données. Durant la deuxième phase, les RBMs sont dépilées pour former l'encodeur et le décodeur (cf. Figure 4). Pour les MLPs, nous utilisons une architecture symétrique multicouche qui prend en entré les images de caractères et apprend directement leur reconstruction. L'apprentissage consiste à mettre à jours les paramètres ou les poids des modèles de façon à minimiser une fonction de coût de reconstruction.

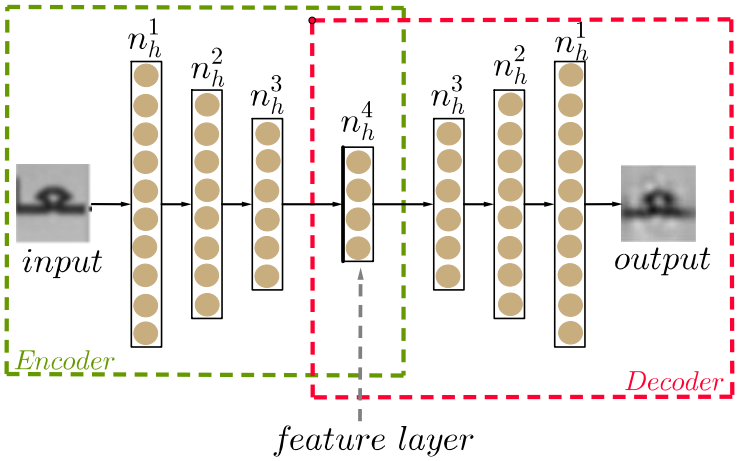


FIGURE 4: Auto-Encodage caractères basé sur le DBN.

Nous proposons, en outre, un deuxième type de modèles de génération de caractéristiques qui se base sur les ConvNets. Ce modèle est entraîné à classifier les images de caractères (reconnaissance de caractères). À l'aide de son architecture multicouche qui apprend l'extraction des caractéristiques et la classification à la fois, le ConvNet est capable de

générer des descripteurs qui visent directement une discrimination entre caractères. En plus, grâce aux concepts de partage des poids, de sous-échantillonnage et de champs réceptifs locaux, les ConvNets permettent la détection des caractéristiques locales et une certaine robustesse aux variations de faible amplitude en translation, rotation et échelle et aux faibles distorsions. Nous utilisons une architecture en six couches. Les deux dernières couches neuronales sont dédiées pour la classification. Une fois le modèle est appris, les sorties de l'avant dernière couche sont gardées comme caractéristiques textuelles. Outres ces modèles qui sont entraînés de façon supervisée, nous proposons un autre modèle de génération de caractéristiques entraînée en mode non-supervisé. L'idée est d'utiliser un ensemble de RBMs qui prennent en entrée des images de caractères et essayent d'explorer les différentes modalités des données couche par couche sans aucune a priori ni étiquetage des données. L'apprentissage se base sur une fonction d'énergie qui contrôle une certaine confabulation entre les unités visibles et cachées de chaque RBM (gestion des probabilités d'activation des unités qui dépend de cette fonction d'énergie). Pour entraîner ce modèle nous utilisons l'algorithme de *Contrastive Divergence*. Chacun de ces modèles est entraîné sur le corpus de caractère de manière à minimiser une fonction objectif bien déterminée qui dépend de la tâche cible (reconstruction, classification, etc.). Dans ces modèles, l'architecture multicouche et les schémas de connections entre elles permettent d'explorer différentes structures multimodales dans les données. La génération des caractéristiques est hiérarchique où des caractéristiques à différents niveaux d'abstraction sont extraites et où tout est appris à partir des exemples (paramètres des caractéristiques, leurs degrés d'importance, leurs combinaisons, leur participation à la classification ou à la reconstruction, etc.). Tout est contrôlé par les poids appris ou en d'autres termes par les données, la cible et le mécanisme d'apprentissage. Il est à noter que pour le corpus d'apprentissage et plus spécifiquement pour le modèle à base de classification, nous ne considérons pas seulement les lettres alphabétiques arabes mais plutôt des formes de lettres. Comme nous avons mentionné précédemment, les lettres arabes peuvent changer de formes selon leurs positions dans le mot. Afin d'améliorer la précision de classification, nous considérons des formes ou des glyphes de caractères. Une fois les modèles sont appris, ils sont appliqués sur les fenêtres glissantes de façon indépendante. Chaque modèle ou chaque type de caractéristiques définit un système d'OCR bien déterminé. La deuxième étape de la reconnaissance consiste à partir de ces séquences de caractéristiques pour apprendre l'étiquetage ou la classification temporelle.

0.4.2 La classification temporelle

Le processus de reconnaissance proposé dans ce travail évite complètement toute phase de segmentation des images de texte. Nous utilisons plutôt un schéma connexionniste récurrent qui apprend la classification temporelle à partir des séquences de caractéristiques apprises. Ce schéma repose sur un réseau de neurones bidirectionnel à longue mémoire à court-terme (BLSTM). Le LSTM [HS97] est un type particulier de réseaux de neurones récurrents où les unités non-linéaires cachées du RNN sont remplacées par des blocs de mémoire avec des connexions récurrentes. Chaque bloc consiste en une ou plusieurs cellules de mémoires. Le flux des informations à travers les unités est contrôlé avec des unités multiplicatives dont les connexions font partie du réseau global ce qui fait que ce contrôle de flux est aussi automatiquement appris à partir des données. Cette structure en blocs de mémoire avec des connexions récurrentes permet au LSTM d'apprendre des tâches de classification des données séquentielles et de

tenir en compte des longs contextes passés. L'architecture bidirectionnelle du LSTM (BLSTM) [GS05, WEG+10] permet , en outre, de prendre en compte le contexte passé et future à la fois pour classifier une observation actuelle. L'aspect bidirectionnel consiste à utiliser deux couches récurrentes cachées connectées ; une parcourt la séquence en entrée dans le sens directe et l'autre dans le sens inverse.

Dans cette thèse, nous utilisons un BLSTM qui prend en entré des séquences de caractéristiques représentant des images de texte. La couche d'entrée est totalement connectée à deux couches récurrentes cachées (chacune traite la séquence d'entrée dans un sens). Ces couches ne sont pas connectées entre elles mais sont connectées à une troisième couche caché non-récurrente qui combine le contexte passé et future. La couche de sortie est une softmax qui produit les probabilités prédites des caractères pour chaque vecteur de caractéristique de la séquence d'entrée.

Pour apprendre cette classification temporelle, le BLSTM doit disposer d'un corpus d'images de texte segmentées afin qu'il reconnaisse l'étiquette ou le caractère cible à chaque instant pendant le parcours de la séquence des caractéristiques. Néanmoins, cette tâche d'annotation de données s'avère très complexe surtout que les caractéristiques peuvent être extraites de façon très fine (scanning avec un pas très petit). Pour remédier à ce problème, nous utilisons le schéma d'apprentissage du BLSTM introduit par Graves et al. [GFGS06, GLF+09, FGS07, GLB+08] pour l'écriture manuscrite en ligne et qui se base sur des exemples non segmentés. Dans ce schéma, le BLSTM est couplé avec un composant de classification temporelle connexionniste (CTC) qui définit une fonction objectif à base d'alignement entre les sorties du LSTM et la transcription cible sans aucune segmentation préalable des données en entrée. Ceci permet ainsi de déterminer les vecteurs d'erreurs temporelles à rétro-propager à travers le réseau. Les composants de ces vecteurs sont définis pour chaque instant et chaque caractère. Le but est d'obtenir des probabilités en sortie du réseaux représentant des pointes qui correspondent bien aux caractères cibles de la transcription et dans le même ordre (cf. Figure 5). Pour se faire, le CTC introduit éventuellement une classe spéciale, outre les classes caractères et dénommée BLANK, qui sera activée lorsque aucun caractère n'est reconnu.

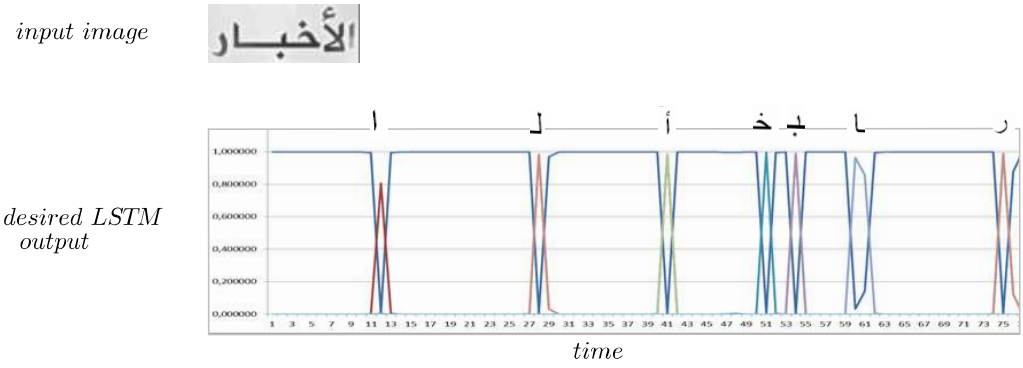


FIGURE 5: Les activations cibles en sortie du BLSTM à travers le temps. Chaque courbe correspond à l'évolution de la réponse du BLSTM pour une classe bien particulière (caractère ou BLANK). Le BLSTM donne, le plus souvent, un BLANK en sortie correspondant à la courbe bleu foncé.

Une fois l'apprentissage est fait, le réseau peut être appliqué sur une image de texte présentée sous forme de séquence d'une des caractéristiques apprises. La reconnaissance

du texte peut être ainsi obtenue en appliquant un algorithme de décodage sur les sorties du BLSTM tel que l'algorithme de *Best Path Decoding*.

0.5 Les modèles de langage

Afin d'améliorer les résultats de reconnaissance, nous proposons enfin d'intégrer l'information linguistique dans le schéma d'OCR. Notre méthodologie diffère des méthodes existantes en deux points: (1) l'utilisation des modèles de langage (LM) appris avec des RNNs aux lieu de LMs fréquentiels et (2) un schéma de décodage conjoint faisant appel aux réponses du LM et d'OCR en parallèle à chaque instant et en partant d'un ensemble vide d'hypothèses, au lieu d'un schéma de classement des réponses d'OCR selon leurs évaluations par le LM. Vu la complexité de la langue arabe, nous proposons d'explorer, dans cette thèse, différents types de LMs (niveau caractères) afin d'améliorer au plus les résultats de reconnaissance. Nous mettons en place des modèles basés sur les n-grams et d'autres plus avancés à base de RNNs. L'idée est de tirer profit de la capacité de ces réseaux à traiter des séquences avec des interdépendances à long contexte qui existent bien évidement dans les textes. Nous construisons une première catégorie de LMs avec des RNNs simples. Le RNN prend en entrée un vecteur concaténant une représentation du vocabulaire (caractères) avec une autre du contexte passé prise à partir des sorties de la couche caché (cf. Figure 6). La couche de sortie du modèle présente une distribution de probabilités du prochain caractère sachant son contexte.

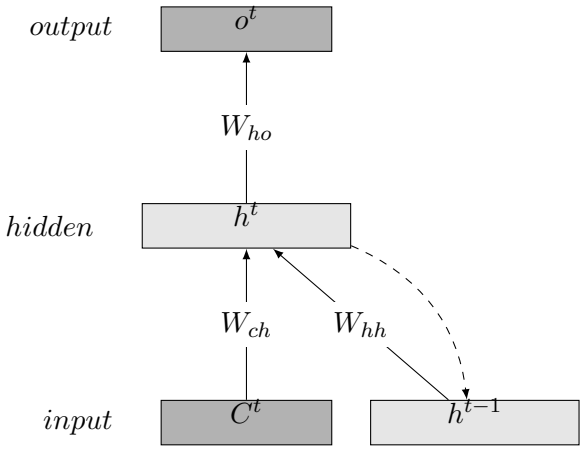


FIGURE 6: Modèle de langage à base de RNN.

Une deuxième catégorie de LMs combine un RNN avec un modèle de *Maximum Entropy* (ME). Les modèles ME peuvent être entraînés avec plusieurs types de caractéristiques y compris celles à base de n-grams. Dans ce travail, nous proposons d'utiliser un modèle ME vu comme un réseau de neurones sans couche caché [Mik12]. Il est entraîné conjointement avec un LM à base de RNN en utilisant l'algorithme de descente du gradient stochastique. Dans cette architecture appelée RNNME, on ajoute des connexions directes entre la couche d'entrée et les sorties du RNN. De cette façon, le modèle RNN se focalise sur l'information complémentaire (à long contexte) à celle apportée par les connexions directes et qui représentent des n-grams basiques. Cette architecture permet

entre autre un apprentissage du LM avec beaucoup moins d'unités cachées que celles requises par un simple RNN. Nous entraînons plusieurs architectures pour chaque catégorie de LM et nous proposons de les comparer avec des modèles n-grams en termes de perplexité et de contribution en OCR. L'apprentissage est fait sur un corpus de texte arabe que nous avons construit et prétraité à partir de plusieurs sources. Concernant le schéma d'intégration des LMs dans la reconnaissance, nous introduisant une version modifiée de l'algorithme de décodage Beam Search. L'idée et d'utiliser à la fois les réponses du LM et d'OCR à chaque étape du décodage pour inférer les probabilités d'extension d'un certain préfixe avec les différents caractères possibles. Le schéma de décodage commence avec un ensemble vide d'hypothèses et construit les extensions pas à pas seulement en utilisant les réponses d'OCR et du LM. Nous introduisons, en outre, dans l'algorithme des hyper-paramètres pour booster les performances de reconnaissance tout en réduisant le temps de réponse. Tout un schéma de redressement des paramètres des LMs, de leurs architectures et des hyper-paramètres de décodage est présenté dans cette thèse.

0.6 Résultats expérimentaux

Comme nous avons déjà mentionné, les différents systèmes de détection et de reconnaissance sont appris et évalués en utilisant des corpus issus principalement des flux TV arabes et qui présentent de larges variations et complexités au niveau des propriétés textuelles, environnementales et au niveau des conditions d'acquisition.

Les trois systèmes de détection ont été évalués sur deux ensembles de test: un présente des critères similaires à celui de l'apprentissage et l'autre venant des sources (chaînes) complètement différentes. Les résultats on montré que les trois systèmes obtiennent des bonnes performances sur les deux ensembles avec des taux de détection qui atteignent les 97% sur le second ensemble. Néanmoins, ces résultats montrent aussi que la méthode à base de ConvNet surpasse les autres méthodes de Boosting notamment en termes de taux de rejet de fausses alarmes. Ceci reflète bien la capacité de classification et de discrimination des ConvNets. À titre d'exemple, sur le premier ensemble contenant 959 zones de texte arabe, seulement 45 fausses alarmes ont été obtenues avec le détecteur à base de ConvNet face à plus que 170 fausses alarmes pour les autres détecteurs.

Concernant la reconnaissance, chaque système issu de l'application de chaque type de caractéristique apprise a été évalué sur les ensembles de texte de la base *ALIF*. Ces systèmes sont dénommés comme suit:

- ***DBN-AE-BLSTM***: DBN auto-encoder + BLSTM.
- ***MLP-AE-BLSTM***: MLP auto-encoder + BLSTM.
- ***ConvNet-BLSTM***: ConvNet Classifier + BLSTM.
- ***Unsupervised-BLSTM***: Unsupervised training with RBMs + BLSTM.

Pour les modèles avec des caractéristiques à base de classification et d'auto-encodage, les taux de reconnaissance caractères (CRR) et mots (WRR) on dépassé 88% et 59% respectivement. Ces taux atteignent même les 94% et 71% avec le modèle *ConvNet-BLSTM*

dépassant ainsi tous les autres systèmes. Ceci est dû au degré de discrimination des caractéristiques utilisées. Pour ce modèle les descripteurs sont appris avec un schéma de classification caractères se qui allège un peu cette tâche pour le BLSTM dans une seconde étape et lui permet de se focaliser le plus sur les interdépendances temporelles. En plus pour les méthodes à base d'auto-encodage, les modèles de génération de caractéristiques sont appris avec un but de reconstruction des caractères ce qui inclut entre autres le bruit, les différentes distorsions et déformations dans les images. Inférer des caractéristiques saillantes pour la reconnaissance est ainsi une tâche très difficile pour les auto-encodeurs. Concernant les résultats du système *Unsupervised-BLSTM*, malgré qu'il sont les moins bons, ils restent assez encourageant pour un modèle qui génère les caractéristiques de façon totalement non-supervisée avec un aspect non-déterministe à une certaine mesure. Ces résultats qui dépassent les 81% en CRR prouvent que les RBMs sont capables d'explorer des représentations succinctes et saillantes des données permettant d'entraîner un LSTM sans atteindre une divergence. Ceci dit que ce système est améliorable si nous tenons en compte plus de masses de données et un réseau plus profond ou une architecture plus large.

Nous comparons aussi ces systèmes à d'autres qui intègrent des caractéristiques conçues manuellement et d'autres non-connexionnistes. Le premier type de descripteurs correspond à des descripteurs géométriques extraits colonne par colonnes après binarisation des images de texte. Le deuxième type de caractéristiques correspond aux Histogrammes de Gradients orientés (HOG). Les résultats obtenus montrent que les méthodes proposées à base de caractéristiques apprises surpassent ces méthodes avec plus de 20 points en termes de WRR et un écart de plus que 39 points pour les caractéristiques HOG. Nos systèmes d'OCR dépassent aussi la solution commerciale ABBYY (l'engin spécialisé en arabe) par à peu près 22 points en termes de WRR.

L'étude de la contribution des LMs est faite sur plusieurs plans expérimentaux qui visent à choisir le modèle optimale et le plus performant et à bien redresser les différents paramètres de décodage. Les meilleurs contributions sont obtenues par les modèles récurrents avec un apport de plus de 15% en WRR dépassant presque par 4 points l'apport des n-grams. Cependant, les contributions des LMs à base de RNN et de RNNME sont assez proches. Les différences majeures entre eux sont liées à la complexité de calcul et la mémoire requise (les RNNME coûtent plus en termes de mémoire alors qu'ils acquièrent moins en temps de réponse par rapport au RNNs). Les résultats finaux montrent qu'avec les LMs, nous arrivons à un taux de reconnaissance mots de 89% dépassant la solution ABBYY de plus que 35 points.

0.7 Conclusion

Dans cette thèse, nous avons attaqué le problème de détection et de reconnaissance de texte arabe incrusté dans les vidéos. À cet effet, nous avons proposé différents systèmes robustes aux complexités du texte arabe, à l'environnement vidéo et aux conditions d'acquisition. Notre première contribution consiste à mettre en place des systèmes de détection de texte spécialisés en script arabe et basés sur différentes méthodes de classification (ConvNet et Boosting). Une étude comparative a été menée entre les systèmes proposés en tenant en compte plusieurs critères de performances. Notre deuxième contribution porte sur la partie reconnaissance où nous avons proposé différents systèmes d'OCR arabe performants qui se passent de toute étape de segmentation ou de prétraitement.

Ces méthodes font appel à des caractéristiques textuelles apprises évitant toutes les incapacités des caractéristiques conçues manuellement au niveau de la représentativité saillante des données. En se basant sur une approche connexionniste récurrente et ces modèles appris de génération de descripteurs, nos systèmes d'OCR surpassent l'un des systèmes commerciaux les plus connus ainsi que ceux basés sur des caractéristiques manuellement conçues. Une autre contribution majeure de ce travail est liée à la construction de plusieurs corpus pour mettre en place et évaluer nos systèmes de reconnaissance et de détection. Ces corpus ont été collectés à partir de plusieurs sources multimédia, principalement des différentes sources TV et ont été manuellement annotés. Face au grand manque d'un tel type de corpus pour la langue arabe, nous avons mis à disposition notre corpus dédié pour la reconnaissance (*ALIF*) pour la communauté de la recherche afin de booster le benchmarking dans ce domaine.

Afin d'améliorer nos résultats de reconnaissance, nous avons proposé d'intégrer, en plus, des modèles de langages avancés à base de RNN. Nous avons introduit un schéma de décodage conjoint qui fait appel aux réponses linguistiques et d'OCR en parallèle. Malgré que nos systèmes de reconnaissance à base de BLSTM aient obtenus des bonnes performances, cette intégration des modèles de langage a apporté une contribution, en plus, de plus de 16 points en taux de reconnaissance mots.

À chaque étape de notre système d'OCR arabe, nous avons mené plusieurs études comparatives afin de tirer des conclusions par rapport aux avantages et inconvénients des méthodes proposées. À l'issu de ces études, nous sommes parvenus à souligner quelques perspectives pour cette présente thèse. Pour les systèmes de détections, une piste d'amélioration possible est d'utiliser la technique de *Multi-Frame Integration* (MFI) permettant de prendre en compte la corrélation temporelle des données et booster la séparation texte/fond. Concernant la partie reconnaissance, les étapes d'extraction des caractéristiques et de classification peuvent être intégrées dans un même schéma d'apprentissage en utilisant le LSTM multidimensionnel [GS09]. Néanmoins, une telle perspective nécessite une grande masse de données annotées et toute une étude du schéma connexionniste pour traiter l'information spatio-temporelle. Pour les modèles de langage, les schémas d'amélioration peuvent encore intégrer d'autres modèles niveau mots et encore des dictionnaires arabes sophistiqués.

Chapter 1

Introduction

The world video library has shown tremendous growth over the past few years. This is mainly due to the increasing number of new TV channels, video hosting services like YouTube and Dailymotion and also due to the early advances in camera devices and storage capabilities. Therefore, structuring and indexing such an amount of data becomes an issue of great importance. It is not only an issue of big data handling but also a need for automatic video understanding. A look at social media like Facebook, Twitter, Snapchat or YouTube, shows that videos become a primary source of information that can be easily loaded and shared like images and texts.

Usually, handling videos is based on the metadata, thus the information that it provides depends on the level of details of the manual annotation. In general, this annotation is a very time consuming task and information is restricted to some elements like dates, creator, few keywords, etc. One solution to these limitations is to use the information within the content itself by automatically recognizing appearing objects, faces and also texts. In particular, embedded or overlaid text in videos is considered as one of the most relevant sources of high level semantic tags used to describe video contents. Usually, it provides information about 'when', 'where' and 'who' elements of the video events and sometimes it gives a verbal description of what is happening and/or said. This is most noticeable in TV Broadcast and becomes widespread in published videos on personal video channels. For example, many tutorial and advertisement videos on YouTube contain text embedded by the creator. This even becomes a trick to increase the audience and many software are developed for this purpose. Therefore, automatically recognizing such texts can provide a large part of needed information for automatic video indexing, chaptering, retrieval, summarizing, search, archiving and many other applications. Usually, This recognition process is referred to as 'video OCR' which stands for video Optical Character Recognition.

Research studies regarding text detection and recognition in videos, have been widely developed during the last two decades. However, they are still dedicated to few languages, mainly to Latin and Chinese languages. For a language such as Arabic, which is used by more than half a billion people in the world, video OCR systems are much less developed. The problem of Arabic text detection and recognition in videos is still not frequently addressed while many great Arabic news channels appeared in the last two decades. The audience and popularity of a channel like Al Jazeera are now comparable

to those of channels like CNN or BBC. According to Socialbakers[1], one of the most popular provider of social media analytic tools and statistics, the MBC Group Arabic YouTube channel is actually one of the top 10 channels in terms of total uploaded video views. Indeed, this lack in Arabic OCR systems is not just about the videos but it also affects natural images. Existing Arabic text detection and recognition solutions are practically limited to scanned documents. All these factors amplify the need of camera-based OCR solutions dedicated to Arabic language. One of the most important needs is to enable the users of this language to take advantages from the applications dedicated to visually impaired people allowing them to 'read' and understand texts existing everywhere [Liu08]. Likewise, this can open new large markets of applications that can help the development of the personal text-to-speech devices or the real time translation of text appearing on videos and images in other languages. Other applications are related to the assistant navigation and applications that facilitate interaction with these contents by avoiding the use of keyboard to manually transcribe what is seen.

In this work, we address the particular task of Arabic embedded text detection and recognition in videos. In addition to the previously mentioned applications and the relevance of the embedded text in videos, our focus on this language is also motivated by the challenges related to the Arabic script in particular. These challenges make always Arabic OCR a very special and an independent research issue. Indeed, Arabic text has many specific properties that make its detection and recognition, in a second step, very challenging. It is a cursive script an morphologically very rich text. In a single word, characters are often connected by a baseline thereby forming sub-words. Moreover, one Arabic character may have more than one shape depending on its position in the word. This text has also different texture characteristics compared to Latin or Chinese ones: more stokes in different directions, different font size and aspect ratio, etc. All these particularities make the application of most of existing OCR systems fail to detect and recognize Arabic Text. One other crucial challenge that faces the Arabic video OCR is related to the datasets availability. In general for Arabic and non-Arabic scripts, there is a noticeable lack of publicly available datasets issued from videos. Unlike scanned documents and even real world images, building a video OCR-based dataset is much more complicated in terms of resources, recording and annotation strategies. Particularly, one additional critical problem for video records is related to the copyright issues specially for videos issued from TV broadcasts and other video hosting services. For Arabic script, in particular, this problem is much more serious. Indeed, the only publicly available Arabic datasets are dedicated to scanned documents like book pages, newspapers, historical documents, etc. However, for Arabic multimedia contents including videos and natural images, there is a complete lack of datasets.

In order to give a clearer idea about the main concern of this work, it is essential to notice that in videos there are two types of texts: scene texts and embedded texts which are illustrated in Figure 1.1. Scene text refers to texts belonging to the scene when recoding the video. Embedded text refers to 'captions' overlaid artificially on videos like names of persons, places, news, etc. Particularly, embedded texts carry important information about the main content and provides relevant metadata for video-based monitoring and structuring services.

In addition to the challenges related to the Arabic text properties, embedded text in videos poses other complexities that make its detection and recognition challenging tasks. A part of these difficulties are related to the environment like the complex background

[1] http://www.socialbakers.com/

(a) Embedded text.

(b) Scene text.

FIGURE 1.1: Embedded and scene texts in videos.

that makes it difficult to discriminate text from non-text regions on one hand and between characters on the other hand. In fact, text embedded in videos can be added not only in boxes with unified background but also directly on the video scene that may include graphics, persons and all possible real world objects displayed in the video. This background can have some structures and textures very similar to texts. Environmental complexities also include some possible visual effects artificially added to the embedded text for ergonomic purposes like varying colors, lighting, occlusions, contrast, etc. Some of these effects are illustrated in Figure 1.2.

Other challenges are related to acquisition conditions of the video like the low resolution, noise, distortions, blurring and degradation. The compression and decompression operations can also degrade the quality of the whole video including embedded texts. This can affect, particularly, character recognition mainly for characters with diacritics.

Our goal in this work is to develop a recognition system that can detect and recognize Arabic texts in videos while being robust to all these challenges related to the text, environment and acquisition conditions. These goals can be defined as two separate steps or integrated in one global operation with many feedbacks. However, they still represent two separate purposes in terms of the OCR system outputs. Text detection aims to determine if there is a text in a given video frame or image and, if there is, to localize it. In that case, the text recognition aims to find the corresponding transcription of the detected text.

FIGURE 1.2: Environmental challenges of embedded texts in videos.

Contributions

In order to reach the previously mentioned objectives, our global approach is defined by two main separated steps: (1) the detection of Arabic embedded text in videos and (2) the recognition of the detected text. Our goal is to propose robust systems that handle these tasks.

The main contributions of this work are:

1. The first contribution is related to the task of Arabic text detection in videos. The global proposed methodology relies on strong text/non-text classifiers. We introduce two classifiers based on a multi-exit boosting cascade. Each one learns a specific type of features for binary classification. A third classifier relies on a Convolution Neural Network (ConvNet) that learns both relevant feature extraction and classification in one connectionist architecture. Both learning procedures and the well-established training data allow to build robust text/non-text classifiers whose outputs are then used by a multi-scale localization procedure of text regions in video frames. The localization procedure is based on clustering text candidates and on a set of geometrical heuristics. We conduct a comparative study between the resulting detectors allowing to draw conclusion regarding the used features and machine learning methods and to highlight the robustness of the ConvNet architecture and its discrimination abilities. The developed methods achieve high detection rates without relying on any pre- or post-processing, like most of the state-of-the-art methods. They take advantage from both a learning procedure and a training dataset to face Arabic text, environmental and acquisition conditions challenges.

2. The second contribution concerns text recognition in videos. Given the cursiveness of Arabic text, we propose a recognition methodology that do not rely on prior character segmentation of the detected text region. This avoids a large part of possible errors that can be caused by over or missed segmentations. Moreover, the recognition scheme do not rely on any pre-processing of the input text images. The recognition problem is seen as a temporal classification task that considers

text images as sequences of features. The sequence labeling or the temporal classification is learned using a Bidirectional Long-Short Term Memory (BLSTM) Recurrent Neural Network. Using the Connectionist Temporal Classification (CTC) objective function, the BLSTM network is trained without pre-segmented data. As for the sequential aspect of text images, we use the multi-scale sliding window technique where a set of features are extracted from the input text image at each scanning step. Given the previously mentioned challenges related to the Arabic text properties, background and acquisition conditions complexity, we propose to learn relevant textual features from data instead of using hand-designed ones. The feature generators are learned separately and once trained, they are applied in each scanning step to transform text images into sequences of learned features.

A first set of features is based on character auto-encoding schemes using different types of deep neural networks, namely Deep Belief Networks (DBN) and Multi-Layer Perceptrons (MLP), trained in a data reconstruction fashion. This leads to two different features generators. The training procedure allows to learn a projection of data into a lower dimensional space while constructing a set of feature detectors through the network layers. A second type of feature generators is based on a ConvNet that learns character classification. While training, the network learns its own feature extractors and classifiers that proved to be robust to geometrical deformations and noise. Once trained, the learned ConvNet in not used as window classifier but only as a feature generator. A third type of features are learned in an unsupervised manner. We use a stack of RBMs trained in a layer wise fashion. This training is totally unsupervised in the sense that it does not target any output. It just explores data modalities through the different RBMs based only on energy flux.

We conduct a comparative study between the different BLSTM-based text recognizers using the different proposed learned features. This comparison allows to draw conclusions about the discriminative capabilities of each model and their robustness to the different challenges. The proposed text recognizers are also compared to other text recognition methods based on hand-crafted features and HOG features in addition to commercial OCR engines. The evaluation and obtained results prove the efficiency of the proposed connectionist learned models and their ability the build relevant features from data.

3. The third contribution is related to the collection of training datasets. One of the main challenges in this work is the complete lack of existing Arabic text datasets issued from videos or natural images. Therefore, we build our own datasets, mainly from different TV Broadcast sources, for Arabic text detection and recognition tasks. The first dataset is dedicated to text detection. It includes a large amount of video frames where Arabic texts are manually annotated in terms of localization. The second dataset is used for text recognition with more than 6 000 Arabic text images issued from videos. This dataset, called *ALIF*, is published for further uses by the OCR community. Another dataset containing Arabic character images is built. All the proposed datasets include a large panel of text challenges (different fonts, colors, sizes, scales, etc.) and acquisition conditions complexity (background, luminosity, contrast, low resolution, etc.). In addition to the efficiency of our learning paradigms in both detection and recognition tasks, these datasets play an essential role in the robustness and the generalization ability of the proposed methods.

4. The forth contribution tackles the integration of the linguistic information in order to enhance text recognition results. Although the proposed methods achieve high recognition results, we proposed to boost them using character-based language models. First, we propose to use Recurrent Neural Network (RNN) based language models that are able to capture long range linguistic dependencies. Second, for the decoding scheme, we are not limited to a n-best rescoring of the OCR hypotheses. Instead, we propose to take advantage from a joint decoding algorithm that uses both OCR and language models probabilities. We introduce a set of hyper parameters to the algorithm in order to boost recognition results and to control the decoding time. The schema is extensively evaluated with different parameters and model architectures. The RNN language models are compared to n-grams and prove higher efficiency. Moreover, the whole proposed scheme allows a considerable enhancement in word recognition results by more than 16 points.

Organization of the manuscript

This rest of this dissertation is organized as follows:

- Chapter 2 reviews existing text detection and recognition methods. We tackle different multimedia contents and present different existing OCR architectures with a special focus on Arabic text.
- Chapter 3 describes different features of the Arabic text and its specificity that may affect its detection and recognition. It presents also the different proposed Arabic datasets (in terms of data challenges, annotations, statistics, etc.) and specifies the used evaluation metrics for text detection and recognition.
- Chapter 4 presents the proposed methods for Arabic text detection in videos.
- Chapters 5 and 6 present our contributions in Arabic text recognition and its enhancement using language modeling.
- Chapter 6 is dedicated to conclusions and gives critical review of the main contributions of this thesis while proposing some tracks of future works.

Chapter 2

Related work

Contents

2.1 Introduction

Text detection and recognition in multimedia contents like photos and videos differ from OCR of scanned document. For the latter, it is almost a solved problem with very high recognition rates. Printed and handwriting texts in scanned documents have been tackled for more than 3 decades and many specific methods have been developed for this type of documents. Nevertheless, for natural images and videos, it is only very recently that it begins to draw the attention of the OCR community. Compared to scanned documents, performances of detection and recognition tasks in multimedia contents are still limited given many challenges related to the environment and acquisition conditions (blurring effects, complex background, non-uniform luminosity, degradation, etc.) and also to the text contents (multilingual aspect, different text aspect ratios, different fonts, etc.).

For scene or embedded texts in videos or in natural images, almost the same detection and recognition schemes and components are usually used. In general, this includes the

pre-processing, detection, localization, segmentation and recognition components that interact in different manners with each other. An additional essential step for videos is the presence of the tracking component. In this chapter, we focus on the detection and recognition steps and we review different existing works. Given the large similarities between video and photo OCRs, our review will incorporate the two fields. We present also works related to Arabic and non-Arabic text detection. Usually, in order to enhance final recognition results, researchers tend to use linguistic information. Although is not an optical component, language modeling becomes an essential step in the OCR schema. A special focus is given to it in this chapter. We present the different existing language models used in the OCR field and we review the different integration paradigms of the linguistic information into the whole recognition scheme.

In the reminder of this chapter, we first give, in Section 2.2, an overview of existing OCR schemes. Section 2.3 presents text detection and localization categories and their related works. In Section 2.4, we review existing text recognition techniques and Section 2.5 is dedicated for language models used in OCR systems.

2.2 OCR schemas: An overview

Usually, in an OCR system, text detection and recognition tasks are considered as two separate modules organized in a pipeline fashion. Recognizing texts in videos or in photos consists, first, in detecting and localizing text lines. Each detected text region is, then, feed-forwarded to the recognition module for transcription. This methodology, known as 'stepwise' methodology and illustrated in Figure 2.1(a), is the most commonly used in existing OCR systems.

Other OCR methodologies consider character classification as the central module that controls both text detection and recognition. There is no pipeline strategy but instead an elementary building of both text detector and recognizer. The general schema of this methodology, known as 'integrated' methodology, is presented in Figure 2.1(b).

2.2.1 Stepwise schema

This methodology presented in Figure 2.1(a), in generally composed of two cascaded main steps: text detection and text recognition. The terminology used in the figure is based almost on the definitions given in [AC99, JKJ04]. Given an image or a video frame, the first step is to determine the presence or not of a text regions. We refer to this step as 'text region detection'. Text localization consists, then, in precising the location of the text by a bounding box that encapsulates the text. Each of the detected text lines passes, then, to the recognition step. The most commonly used recognition techniques are based on a segmentation step of the text line into characters. Each character is recognized separately and results are concatenated to produce the final transcription. For some other methods, the text recognition is segmentation-free in the sense that the text line is sequentially recognized using temporal classification methods without relying on a character-based segmentation step. We will present and review these recognition methodologies in details in Section 2.4.

Stepwise schema has been widely used for text detection and recognition in videos [SKHS98, HYZ02, Che03, Lie03, PBQT12, MBHV12, EGMS14] and in images [BCNN13, YZB+13,

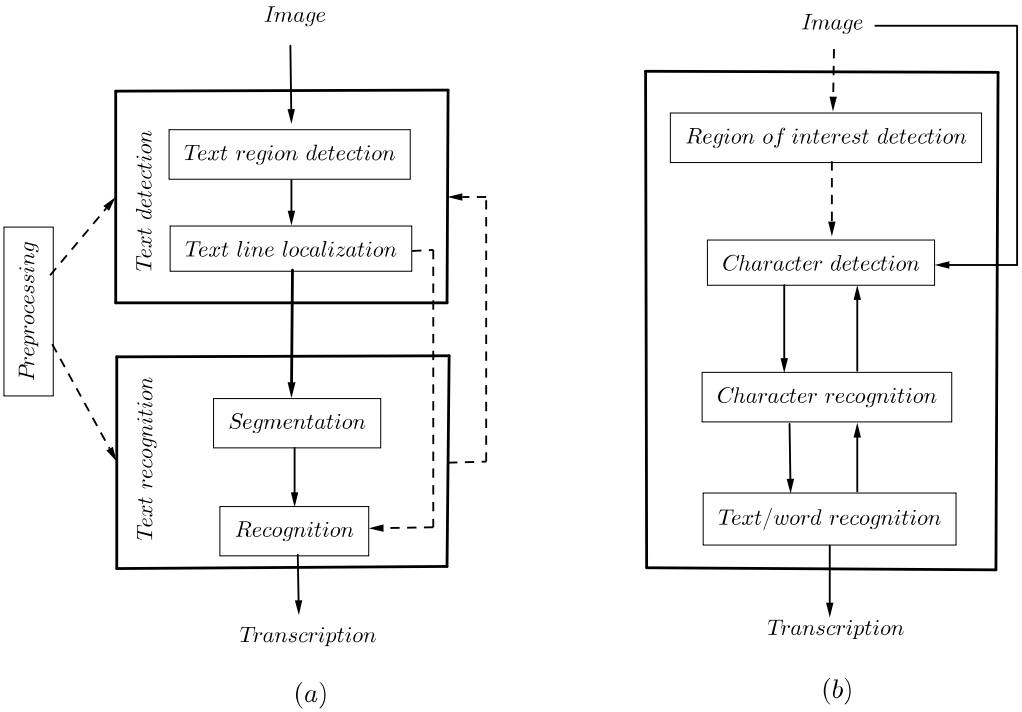


FIGURE 2.1: Text detection and recognition methodologies: (a) stepwise schema and (b) integrated schema.

YBL+12]. For end-to-end video text recognition, a tracking step is usually added after detection in order to tackle the whole recognition chain starting from the input video and ending at the transcription results. We note that for these methods, the recognition step can have a feedback to the detection module. In [JSVZ16], the information gained from word recognition is used to rank and merge detections.

This stepwise schema includes the basic modules used in text detection and recognition processes. However, many pre-processing and post-processing steps that they differ from one method to another can be added. For instance, in [MBHV12], text detection encapsulates a pre-processing module consisting in a multiple frame integration (MFI) [YPX09, HYZ02] method to minimize the variation of the video frames background. It includes also a binarization step of the whole frame for better detection and to facilitate a word-based segmentation of the extracted text. Besides, a lexicon-based verification is used as a post-processing step in order to correct some recognition errors.

In [EGMS14], the problem of Latin text recognition in multimedia documents has been addressed. The authors have used an off-the-shelf text line detector and considered two methods of text recognition: one is based on character segmentation followed by character classification using ConvNets and the other is segmentation-free using sliding windows and graph-based labeling. In order to enhance recognition results, a post-processing step using n-gram language models is added.

The separation between pre-processing, detection and recognition steps in such schema makes easier the focus on each module separately. This can be very efficient given that problems related to each module are, almost, filtered one by one. For the text

recognition step, it is much easier to handle one text patch already extracted rather than the whole text image or frame with all background variations [YZB+13]. Moreover, the stepwise procedure allows to easily handle rotated texts since the re-orientation can be performed before the recognition step and the orientation angle can be estimated within the detection step. Another advantage is that separating modules makes possible to use off-the-shelf detection or recognition blocks. However, for this type of methodologies, the most important problem remains in the luck of effective dialogue between different modules which may produce some error accumulations through stages.

2.2.2 Integrated schema

Unlike stepwise methods, the integrated schema is a bottom-up methodology for both detection and recognition. The idea is to focus first on characters, as elementary units of the text. Thus, this schema starts directly with the character classification step that is applied in many locations of the image in an attempt to not only separate characters from the background but also to separate characters from each other. The classification information is then mutually used to construct text components for the detection and localization and to recognize the text. This methodology is more frequently used for scene texts in natural images where a text is limited to few words or to one word by line. Such a schema depends strongly on the accuracy of the character classifier and on the interactions rules between this classifier and the detection and recognition modules.

For Arabic text, as a cursive script, it is highly difficult to apply this methodology. Arabic is a morphologically rich script with high number of character classes (or character glyphs). Separating characters from the background and from each other is, hence, much more complex compared at least with Latin script. It may induce additional earlier problems, especially for unconstrained text recognition. Moreover, it is a time consuming process if we consider a large lexicon and text lines with variable lengths. Integrated methodologies are more suitable for word spotting tasks.

The integrated methodology has been applied in [NM13b] for unconstrained end-to-end text recognition in images. In this work, a set of convolutions have been applied on the image gradient field using a set of oriented bar filters which generates strokes. Characters are, then, detected and recognized as image regions containing specific strokes in terms of orientations and relative positions (cf. Figure 2.2). Using a nearest-neighbor classifier, potential character candidates are selected among the obtained regions. In other work [NM13a], the same authors proposed not to reject non-potential character segmentations but instead to keep multiple segmentations of each character until the last stage of the processing when the context of each character in a text line is known. This allows enlarging the set of possible recognition hypotheses. Recognition is performed using a dynamic programming algorithm applied on a directed graph formed by character classification and linguistic scores.

In [WWCN12], a ConvNet character/non-character classifier has been proposed and applied in a multi-scale sliding window schema in order to look for character candidates. These candidates are then classified and characters are recognized using another multi-class ConvNet classifier. Finally, using a beam search algorithm, the outputs from the text detection and character recognition modules are combined to obtain the final end-to-end word recognitions. In [WB10], however, the character classifier has been directly applied at each position of the image in order to both detect and recognize character

FIGURE 2.2: Examples of character-based candidate regions obtained in [NM13b]. Top image corresponds to bounding boxes of strokes.

regions. The classifier is based on histograms of oriented gradients (HOG) and nearest neighbor algorithm. Once characters are identified, word recognition is based on pictorial structure framework [FE73] that takes as inputs the locations and scores of detected characters and finds an optimal configuration of a particular word (cf. Figure 2.3).

FIGURE 2.3: Text detection and recognition in [WB10]. Characters recognized using HOG features and merged into words using pictorial image.

2.3 Text detection in multimedia documents

Text detection in multimedia documents aims at determining the presence or not of text regions and at precisely localizing it by limiting it using a bounding boxes for example. In a stepwise recognition procedure, the goal of this stage is to extract the text from the image or the frame in order to recognize it. For some integrated methodologies, a part of this step may be applied to detect regions of interest without a precise localization. This latter can be then performed on a smaller space using character classifications. Many works have addressed the problem of text detection in videos and camera-based images. They often try to find specific features of the text that characterize it from the background like edges, gradients, color, texture features, point and region features, etc.

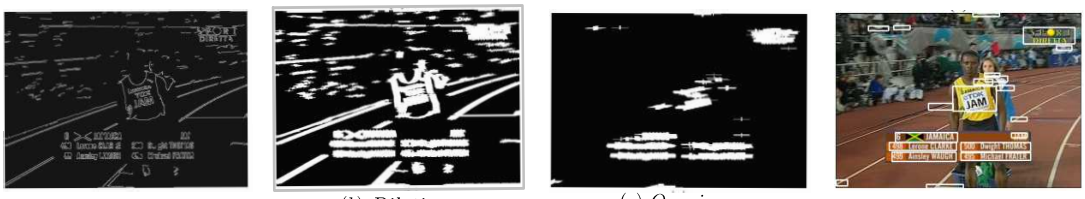
These features are used in different manners. They can be used by machine learning methods. Heuristic rules can also be applied on these features in a way to find a better representative model of text regions. They are also used by hybrid techniques that combine machine learning methods and heuristic rules. In this section, we review existing text detection methods used for both video and photo contents.

Although text detection in these multimedia documents seems to have similar challenges, each of the video and image contents has, in reality, some minimalistic specifications that may affect the detection results. For texts in videos, main challenges are related to the low resolution, noise, and compression artifacts. This can add distortions to character edges and affects text strokes. In particular, for Arabic script, this can be very problematic given the strong presence of small structures that describe characters like diacritics in addition to the delicate strokes in most characters extremities. However, these limitations do not negate the existence of other challenges related to occlusions, lighting, etc. For texts in natural images, most common limitations are related to the conditions of the photo capturing that can involve non-uniform luminosity, image blurring and perspective distortion.

2.3.1 Heuristic-based methods

Derived from scanned document analysis research, heuristic-based methods apply a set of manually inferred rules directly on the low-level features based on the observation of text characteristics. Some of them exploit text edge properties such as distribution, density and strength. They are also known as Connected Component (CC) based methods. These latter can be seen as sets of labeled features grouped by rules of similarities i.e. close colors, edges, corners, etc. In order to define text regions, heuristic rules are applied to find feature and spacial consensus among these components.

Anthimopoulos et al. [AGP07] created an edge map with the Canny edge detector. Text regions are constructed by applying morphological operations (dilation and opening) on edges, as presented in Figure 2.4, which create region-based CC. Using some distance heuristics applied on these components, an initial set of text regions is produced. In order to increase the precision of the bounding boxes, horizontal and vertical edge projections of each box are performed. Using some thresholds, some boxes are discarded and some split and merge cases are rectified. Almost the same methodology has been applied for Farsi/Arabic text detection in videos [MMO10].



(a) *Edge map* (b) *Dilation* (c) *Opening* (d) *CC analysis*
Initial bounding boxes

FIGURE 2.4: Text area detection used in [AGP07].

Phan et al. [PST09] used the Laplacian operator followed by maximum gradient difference value computation. Text regions are then constructed using the k-means algorithm. Edge and gradient features have been also used, in many other works, for text detection in videos [SHT08, XHC+01, CSL02, LSC05, KK09] based on Sobel or Canny edge map and also in natural images [PLK+10, BMB13, HSU13]. In fact, strong and symmetric gradients have been always considered as relevant features of texts regarding the background.

Other works are based on color features in order to separate text regions and background [WK03, YT11]. In [YT11], a structural analysis is performed from text characters to text strings. First, CCs are constructed based on candidate text characters distinguished by color and gradient features. Texts are then localized using adjacent character grouping and text line grouping heuristics. The proposed method outperformed the state-of-the-art results on the public Robust Reading Dataset.

Other heuristic approaches consider texture features to characterize text regions and form CCs. Although texture features, like Discrete Cosinus Transform (DCT), Wavelet, LBP and HOG are mostly used with machine learning-based methods, they have been applied with heuristic approaches. In [KK09], heuristic rules based on different LBP features have been used to refine and verify text regions initially selected based on edge features. In [CAK03, LCL00], DCT coefficients of intensity images have been integrated as text features within heuristic-based methods.

Epshtein et al. [EOW10] proposed a novel text descriptor named the stroke width transform (SWT) which quantifies the stroke width for each image pixel. The operator computes for each pixel the width of the most likely stroke that contains the pixel. By measuring the width variance of each component, texts can be easily extracted since text is characterized by fixed stroke width. The method has been applied on scene text detection in images and provided great results in ICDAR 2003 and ICDAR 2005 text detection competitions both in terms of speed and robustness versus font and language variations. However, the operator performs poorly in images with high noise and illumination variations.

Heuristic-based techniques go back to first attempts to handle text detection in multimedia document, when they were very adequate to text in scanned documents. Their use was then very intuitive in addition to the fact that they are visually interpretable. This makes very easy their tuning to fit some specific contents. However, heuristic-based techniques are very sensitive to variable background and image quality. Therefore, they are not very effective in detecting texts in videos. Moreover, they heavily rely on manually setting a set of parameters, which makes their performance very dependent on the processed data.

2.3.2 Machine learning-based methods

Many machine learning-based methods have been proposed for text detection. They aim to learn discriminative features from a training data in order to build a text/non-text classifier. For text localization, these approaches usually use a sliding window technique. The classifier scans the image at different positions and scales producing text candidates. A grouping algorithm is then applied to produce text regions.

Delakis and Garcia [DG08] proposed a robust system for horizontal text detection that is based on a ConvNet applied on both images and video frames. A pipeline of convolution and sub-sampling layers captures textual features and passes them to a Multi-Layer Perceptron for binary classification (text/non-text). The system is strongly inspired from their work on face detection in images [GD04]. The proposed method outperformed other methods on the real-world test set of ICDAR'03.

Kim et al. [SKPK00] method relies on Support Vector Machine (SVM) in order to directly classify video pixels without prior knowledge. Another method from Li et al. [LDK00] is based on a Multi-Layer Perceptron (MLP) that learns mean, second- and third-order moments of the image wavelet decomposition. A sliding window technique is then applied on video frames where the decomposition and the feature extraction operations are performed and the MLP classifies frame regions. In [LLL+11], the authors used Adaboost to classify image windows that are presented as set of features like local energy of Gabor filter, statistical texture measure of image histogram, measurement of variance of wavelet coefficient, etc. The trained classifier is applied in a multi-scale scanning scheme of the input image using sliding windows. Zhang at al. [ZGCZ05] proposed a method for video text detection based on LBP features extracted from sliding window in a multi-scale scanning scheme and cascade histogram matching.

A detection method has been proposed in [CCC+11] for texts in scene images based on unsupervised feature learning. Using character training images, a set of feature vectors are learned using k-means algorithm. The learned features are used to describe sliding windows that sweep input images where a character/non character classification is performed using a learned linear SVM.

Machine learning based methods proved good discriminative and generalization abilities in many existing works. However, given that they are strongly related to the sliding window technique, they can be, in some cases, very time consuming. Usually, machine learning-based methods are used jointly with heuristic-based methods resulting in what we call 'hybrid methods'.

2.3.3 Hybrid methods

Another class of hybrid methods that makes use of both heuristics and machine learning techniques has been also proposed. In general, a coarse text detection is first performed using heuristic rules. Then, a feedback pass takes place to reject false alarms using machine learning-based techniques. The resulting schema can then take advantage from both methods and allows using the sliding window technique while reducing time complexity.

Ye et al. [YHGZ05] proposed a coarse-to-fine procedure for image and video text detection. In the first pass, a wavelet energy-based decomposition is performed. It is followed by a density-based region growing method that connects the resulting text pixels. In the second pass, a texture features classification is applied on the obtained text lines using SVM for accurate text identification. Anthimopoulos et al. [AGP10] proposed a two-stage schema for video text detection. Text line regions are determined using edge filters and some heuristic rules (dilation, smoothing, projections etc.). Then, obtained results are refined by a SVM classification based on edge Local Binary Patterns (eLBP).

In hybrid methods, the machine learning-based pass is not only used to refine text localization results. It performs also detection and rejection operations in lower dimensional space. In [SSP+12], the authors used Laplacian and Sobel operators to enhance low contrast text pixels in input video frames. They applied then a bayesian classifier to classify true text pixels from the enhanced text matrix. Using boundary growing method based on the nearest neighbor concept, the method can handle multi-oriented texts in video frames.

Conditional Random Fields (CRF) have been successfully used to detect and localize Latin text in TV Broadcast [PCPN11]. Text regions are first extracted using edge features and an SVM that predicts text/non-text classification. The CRF is used for text line aggregation and localization. It takes as features the estimated posterior from the SVM where a distance-based function is used to compute optimal labels for text regions.

In [HQT14], a hybrid method has been proposed for text detection in natural images combining Maximally Stable Extremal Regions (MSERs) [NS08] and ConvNets. The idea, illustrated in Figure 2.5, is to first apply MSERs detector on the input image in order to generate text components. A trained ConvNet classifier is then used to assign confidence value to each MSER component. A threshold on resulting confidence values gives final text-lines. In order to correct some error connections between multiple characters produced by the MSER, a sliding window with the ConvNet classifier is applied. The method has been evaluated on the ICDAR 2011 benchmark dataset and outperformed existing methods.

FIGURE 2.5: Text detection method used in [HQT14]. (a) Original image, (b) MSER component candidates, (c) component confidence map after applying ConvNet and (d) final detection after a simple thresholding.

In [ZFLW15], a hybrid method, composed of two stages, has been also proposed for scene text detection in natural images. In a top-down stage, a learning-based Partial Differential Equations (PDEs) system is first learned off-line with L1-norm regularization on training images. Applied on a text image, a high quality text confidence map is produced. Using an additional local binarization step on these confidence maps, text region candidates are then extracted. In the bottom-up stage, character candidates are detected based on their color similarity and then grouped into text candidates by simple rules.

2.4 Text recognition in multimedia documents

Compared to text detection, text recognition is more sensitive to the type of the processed document, the acquisition conditions and the processed text. As we previously mentioned, there are much more advances for text recognition in scanned documents with both printed and handwriting texts. Moreover, most existing works handle a restricted circle of languages like Latin and Chinese. Handling text recognition for Arabic script and for camera-based documents is limited to one or two works and even non existent for natural images. Thus, in the first part of this section, we mainly review recognition methods for the Arabic script in scanned documents and for printed text given that these conditions are closer to our task.

In general, for Arabic or non-Arabic scripts, the focus on text recognition in camera-based documents has began with a direct application of methods developed for scanned documents with unified simple background and relatively less challenging acquisition conditions. Thus, first attempts tried to make similar environment, by performing pre-processing on images and video frames, in order to simply apply scanned document recognition methods. Most common pre-processing steps are related to binarization. We present, in this section, existing works in this field. Concerning the recognition task, many methodologies have been proposed, tackling the problem in different manners for image and video documents. Some methods resort to an explicit segmentation of the text into characters where each character is recognized separately. Other methods are segmentation-free where the text is holistically or sequentially recognized. In parallel, many surveys have been published for OCR systems in general. Most of them addressed text recognition in scanned documents [MEA12, LG06, PM13, Vin02, Sin13] and to a less extent text recognition in natural images [LDL05, ZYB16, YD15, CWW15] and in videos [Lie03, SPB12].

2.4.1 Arabic text recognition

Methods

Like Latin-based OCR techniques, existing Arabic printed text recognizers mostly resort to an explicit segmentation of the text image into characters. In [MBHV12], an Arabic video OCR system has been proposed where the authors tended to use many pre-processing steps in order to simplify the recognition task. Since the text extraction step, a binarization is applied to video frames using pixel clustering and text pixels are enhanced using Gabor filter. The video OCR problem is then reduced to tackle recognition of black text on white background. Text lines are segmented into words. For recognition, the authors considered two parts in the structure of Arabic words: the body of the word and the diacritics (cf. Section 3.2). Diacritics are filtered out using geometric heuristics and the word body is segmented into characters and sub-characters based on pixel matching. This procedure is illustrated in Figure 2.6. A set of hand-designed features are extracted for each segment based on black and white pixels transitions, diacritic positions, horizontal and vertical profile projections, etc. Each segment is then classified using fuzzy k-nearest neighbor algorithm. Although this OCR system got high recognition rates, the proposed method depends strongly on the pre-processing step which seems to be very well tuned to the processed video frames.

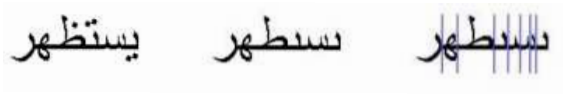


FIGURE 2.6: Segmentation step of the Arabic text recognizer proposed in [MBHV12].

Similarly, a segmentation-based method for printed Arabic text recognition has been proposed in [Sha08] where character segments, presented by moments invariants and SVD features, are classified using neural networks. In [AAKM+12], printed text has been segmented into words, sub-words and characters and segment classification has been performed using template matching. Another Persian/Arabic text recognizer has been proposed in [MPR05] where segment classification is based on a MLP/SVM combination.

The segmentation step has been considered the basis step in Arabic text recognition for many decades and several segmentation approaches have been applied in this context. This step is very critical for this type of script given its cursiveness and the presence of ligatures (cf. Section 3.2). Character classification techniques do not differ very much from those used in non-Arabic scripts which is not the case for segmentation techniques. One of the first attempted methods used in Arabic character segmentation is based on vertical projection (histogram) [MNY99, ZSSJ05]. Other segmentation techniques are based on contour tracing [RPML95, PLDW06, MPR05] which allows to determine touching points between characters and end of characters. Thinning approaches that determine the skelton of the text are also applied for character segmentation [TSAA93] in addition to the graph theory [XPD06, Zid04], morphological operators [TF96] and template matching [BS97].

Although it is intuitive, segmentation-based recognition is an error-prone specially for cursive script with overlapping and ligature cases [Kho02] as well as for texts with complex background. Other segmentation-free OCR systems have been proposed [PCS+13]. Some of these approaches tend to sequentially analyze text line features generated using most often a sliding window. Based on a character model, each observed subset of features is probabilistically classified as a specific character without any prior knowledge about character boundaries. In [AMMQ08], this technique is used and a set of 16 features, based on the sum of black pixels per strip, is extracted for each window. The resulting feature vectors are then fed to a Hidden Markov Model (HMM) for character sequence labeling. In [ARFM13], sub-character HMM models have been considered in order to take into account common patterns between different characters as well as between different shapes of the same character. An example of this model is illustrated in Figure 2.7 for the letter 'Siin' (س).

Recently, many works have been proposed that take advantage from recent advances in temporal classification to perform printed Arabic text recognition with Recurrent Neural Networks (RNNs) [RSRvdN13, UHBAR+13]. Indeed, the approach has been successfully applied for Arabic handwriting text recognition and proved state-of-the-art results in many competitions in this field [Gra12a]. The use of RNNs for Arabic as a cursive script and in scanned documents allows to completely overcome all kinds of segmentation and any HMM-based infrastructure including character or sub-character models. The system can simply take column-wise text features as inputs and learn transcription as a feature sequence labeling problem.

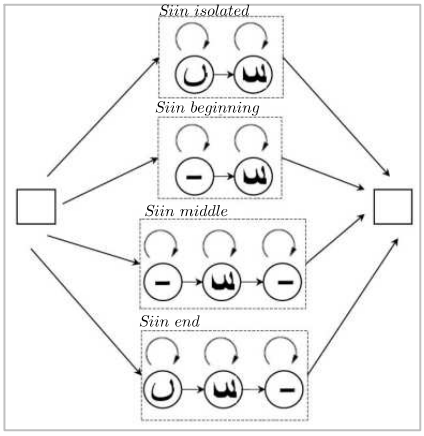


FIGURE 2.7: HMM structure for Arabic character 'Siin' using the sub-character model proposed in [ARFM13].

Another issue that has been deeply tackled for Arabic text recognition is the font recognition. In fact, Arabic printed script includes huge range of variations in fonts which strongly affects recognition rates. Some works even tend to handle mono-font Arabic text recognition [NUS+15]. Other works propose Arabic font recognition [SKH+13, LMA15] and even a competition has been held within ICDAR conference for multi-font Arabic printed text recognition [SKEA+11] (cf. Figure 2.8).

FIGURE 2.8: Font variation in Arabic printed text tackled in [SKH+13].

Arabic Datasets

As previously mentioned, existing Arabic text datasets are largely dedicated to scanned printed or handwritten documents. The main target application is to digitize Arabic books and textual documents so that they can be searchable.

The first Arabic text datasets address handwritten documents [PMM+02, AOCS00, MKKA12, MCA+10]. They deal with both historical and modern Arabic manuscripts. As for printed text, the focus has been made on scanned papers and computer generated documents, where the text is usually well structured. Many datasets have been proposed in this context. For example, the DARPA Arabic text dataset [DH97] has been created

from magazine articles, book chapters and computer generated documents with 345 page images in total. The dataset was created by the US Department of Defense and it is not freely available.

Another dataset, called Arabic Printed Text Image database (APTI) [SKA+13], has been created in order to evaluate Arabic text recognition in screen captures or in images extracted from PDF documents. It is composed of 45,313,600 Arabic printed word images, with 10 fonts, 10 sizes and 4 different styles (italic, bold, etc.). APTI is a very large dataset, but it is made up of synthetic text images with a clean white background. In the same context, the ATID/MF database (Arabic Printed Text Image Database/Multi-Font) [JKK+13] was built from 387 pages of Arabic printed documents scanned with a grayscale format. Pages are extracted from the official website of a Tunisian newspaper.

2.4.2 Text binarization

Binarization has been considered one of the most crucial steps for text recognition in natural images and videos specially for segmentation-based methods. It simplifies and enhances both text line and character segmentation while eliminating all complexities related to the background. Existing methods are usually related to adaptive thresholding, clustering and probability models.

Adaptive thresholding
Adaptive thresholding calculates binarization threshold in pixel local neighborhood of each pixel [SP00, Ber86, Nib85]. It is very suitable for large variation in the background intensity. It fixes many problems related to global thresholding like in the Otsu method [Ots75] which try to find a single threshold value for the whole document.

Many works have used adaptive binarization as a pre-processing in video or natural images text recognition [LW02, ZLT10, LSC05, YGH04, NGP11]. In [ZLT10], the method, dedicated to video texts, is based on edge detection. Then, it fills up it using local thresholding to decide the inner side of the contour. In [NGP11], the authors detected the baseline in order to determine the main body of the text. Then, the stroke width of characters is detected and an adaptive thresholding is applied followed by an enhancement step of the binarization output using the convex hulls information. The method has significantly improved video OCR results.

Clustering
Other image and video text recognition techniques use binarization based on clustering or, in general, on machine learning approaches. The idea is to consider text binarization as a segmentation problem where text pixels are presented as a cluster separated from the background pixels. These methods proved to be more suitable for video contents and also natural images with blurring effects, varying luminosity and complex background sometimes similar to the text texture.

In [WK11], the authors proposed a binarization technique that combines color-based clustering and SVM character classification. The input image is first tentatively binarized using k-means clustering in the HST color space. Then, the obtained binarized images are segmented into 'single-character like' images where a SVM is used to classify each image into character or non-character. The binarized image with maximum SVM responses is selected as the final result.

Saïdane et al. [SG07b] proposed a Convolutional Text Binarizer (CTB) based on a ConvNet trained in a supervised manner. It takes as input color text patches and learns to output the corresponding binary image where text pixels are set to 0 and background pixels to 255. The proposed CTB has been applied to video texts and proved to enhance video OCR without using any tunable parameters.

Probabilistic binarization
A probabilistic binarization model has been proposed in [YGH04] where Gaussian Mixture Models (GMMs) of intensity and hue components in HSI color space is trained on a set of sampled pixels. The pixels are selected using a rule-based sampling under the assumption that the text pixel always lies between an 'edge couple'. A combination of the GMM with spatial connectivity information of the character strokes allows to build final binarized images.

A benchmark of various binarization methods on ICDAR 2003 and ICDAR 2011 natural images has been presented in [MBN+13]. Text recognition accuracy has been evaluated for different binarization techniques and state-of-the-art results were obtained with local non-linear Niblack.

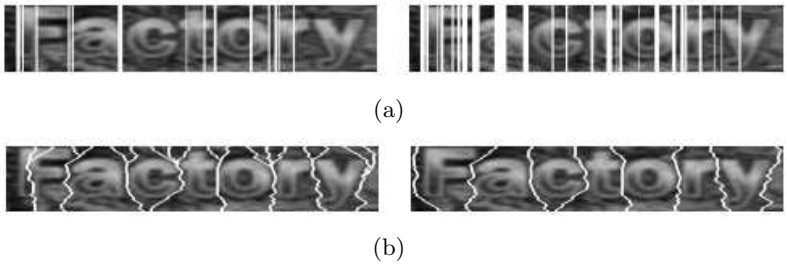
2.4.3 Text recognition

As presented in Section 2.2, the whole detection/recognition schema can be stepwise or integrated and strongly affects the recognition methodology. For a stepwise schema, once the text has been detected, its recognition can be based on a prior segmentation into characters where each segment is recognized separately or can be holistic considering the text image as one single pattern or as sequence of features. Thus, recognition methods under this schema are classified into segmentation-based or segmentation-free methods. Another class of methods, which is very related to the integrated schema, consider an end-to-end recognition process where we cannot separate detection, holistic recognition, character recognition, language model integration from each others. The recognition component is defined by the interaction of all these operations and can be refereed to character or word spotting process.

Segmentation-based recognition
Once detected and extracted, text images, under this recognition methodology, are segmented into characters. We note that for segmentation-based methods, no recognition is done until the text region is fully segmented. Usually, segmentation is achieved using a vertical profile projection analysis [LW02]. Other methods propose to boost results using heuristic rules to further split and merge obtained segments relying on assumptions about the characters' widths and heights [MZJ+07, HMZ09].

However, for video and natural images with a relatively complex background, this method can lead to over-segmentation or missed segmentation cases (cf. Figure 2.9(a)). Finding a sophisticated segmentation means finding the optimal threshold related to the used projection which is very critical for such contents and leads to recognition errors. This problem has been deeply tackled in [PSST11] where the authors proposed a method based on gradient vector flow for video character segmentation. The method operates directly on grayscale video text regions without any prior binarization. It consists in finding curved segmentation paths corresponding to candidate cut pixels by formulating character segmentation as a minimum cost path finding problem. The procedure has

been applied within a two-pass path finding process where, in the forward pass, potential cuts are located. The backward pass serves as a verification step in order to reject false cuts as precised in Figure 2.9(b).



(a)

(b)

FIGURE 2.9: Character segmentation with both profile projection analysis (a) and the method based on gradient vector flow proposed in [PSST11] (b). (a) illustrates missed segmentation (left) and over-segmentation (right). (b) presents the forward pass (left) and backward pass (right) results of the proposed path finding method.

A learning-based character segmentation method has been proposed by Saïdane and Garcia in [SG08]. They used a ConvNet that takes as input color text patches and learns to output a vector that classifies input columns into border or character zone. The network is then trained in a supervised manner using synthetic text images where the exact positions of the different characters are known. The method has been tested on both synthetic data and text images issued from videos and web pages, providing high segmentation accuracy. However, it is dedicated to vertical cuts and handling overlapping cases is not guaranteed.

Text recognition for this type of methods consists in a character-based classification of each segment. Classifications are then concatenated in different manners to form the final transcription. In [SGD09] and as a continuity of their work [SG08], Saïdane and Garcia proposed a text recognition scheme that uses the ConvNet-based segmentation results. However, here, an over-segmentation is performed by using a lower threshold on the local maxima probabilities. The used character recognizer is based, similarly, on a ConvNet trained to classify character images [SG07a]. The recognizer is applied on each segment. The text recognition scheme is based on a weighted directed acyclic graph where neighboring segmentation candidates are connected and character recognition results represent connections' weights. Recognition is performed using best path search algorithm which keeps only correct segmentation.

In [EGS11, EGMS14], almost the same video text recognition paradigm has been used. However, the character segmentation method is based on a shortest path algorithm proposed in [LLP96]. The segmentation has been performed on processed text images where the background has been separated beforehand from text pixels using a combination of intensity analysis and multi-frame integration.

Bissacco et al. [BCNN13] proposed a complete OCR system, Google PhotoOCR, that performs all the recognition chain from detection to transcription incorporating also language models. The system takes advantages from recent advances in deep learning and is based on prior segmentation of text images after their detection. The recognition step relies on a prior over-segmentation of the text image using the combination of two

methods. First text images are binarized and using some morphological operations, connected components are extracted. Then, a binary sliding widow is applied not to identify characters but to detect segmentation points between characters. Each window is described by a HOG feature vector combined to Weighted Direction Code Histogram features. A binary logistic classifier is trained on these features and then evaluated on the sliding window when sweeping the text image. This procedure gives as output a vector of the positions of the detected segmentation points. For character classification, a deep neural network is trained using HOG features. The learned classifier is applied on the obtained segments. A beam search algorithm [RN95] is used to determine optimal segmentation points which maximizes a score function of classification responses and linguistic information. The main goal in this approach is to find the good segmentation points.

Segmentation-free recognition

Segmentation-free text recognition regroups methods that do not apply any prior *explicit* segmentation of the detected text image into characters (or elementary text units in general). The motivation behind these methods is to avoid all errors that may be caused by the segmentation step. Most segmentation-free methods use the sliding window technique. Given a text image, the goal is not to find character boundaries but, instead, to identify characters directly while sweeping the image or to sequentially extract features from it that can be further temporally classified. Other methods resort to a direct character identification within text regions. This step does not aim to only segment the text but it is a part of the text recognition scheme. Indeed, the results of these techniques are further integrated in more global recognition scheme where the character identification results or the partial feature representations are used to infer transcription using optimization methods like CRFs, HMMs, RNNs, graph models, etc.

Roy et al. [RRS+13] proposed a method for multi-oriented text recognition in natural scene images based on a sliding window with an estimated path. Text images are first binarized and the window trajectory is estimated by a polynomial function (cf. Figure 2.10). Then, local gradient histogram and Marti-Bunke features are extracted from each sliding window. The text image is then presented as a sequence of hand-designed features that feed a HMM to learn sequence labeling. Using the HTK toolkit, the HMM learns first character models. Then, given the transcription of each text line, character models are concatenated to form text line model. The recognition or transcription is performed using the Viterbi algorithm.

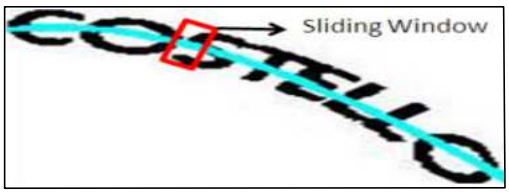


FIGURE 2.10: Sliding window with estimated path used in [RRS+13].

HMM recognizer has been also used in [SCS09] for sliding text recognition in Turkish broadcast news. In this work, the authors use the thinnest sliding window and represent bitmap values of each column in the grey-level of the extracted text by a 20-dimensional vector. The sequential representation of the text is then given as input to an HMM

decoder in order to perform text recognition. Single Gaussians are used to represent the output distributions of HMM states. In order to train different glyph models, a set of synthetic character glyphs are used to construct a training data from video texts. Training and recognition have been performed using the AT&T tools.

In [MAJ12b], windows at multiple scales and spatial locations are used to detect characters within text regions. The character classifier applied for each window is based on a multi-class SVM that learns classification from HOG features of character images. Text recognition is performed using a Conditional Random Field (CRF) on the character detections by connecting them with edges. The CRF jointly models the strength of the detections and interactions among them.

In [EGMS14], a multi-scale scanning scheme is applied on video text images using a set of windows with different aspect ratios. A character classifier based on a ConvNet is then applied on each window in order to identify each character presented in the window or reject it as non-valid one. Similarly to [SGD09], the text recognizer is based on graph model used to represent the spatial constraints between different overlapping windows. A Viterbi algorithm is then applied to determine the best sequence of characters corresponding to the text image.

Other segmentation-free methods are based on direct character identification or detection within the text region. It seems to be a segmentation-based approach but here, in contrast to these latter, the recognition process begins from this character localization and identification. This partly recognition information is further integrated in a global recognition scheme of the whole text and not used just to determine character boundaries. In [SWX+13], character candidates are identified using a Part-based Tree structured Model. One model is assigned for each character using a tree-based representation where nodes present parts of the character and edges model the topological relations of nodes. HOG features are used as local appearance descriptors and the overall model is learned in a supervised manner to model each character. Figure 2.11(a) presents an example of the character detection and recognition results. Based on these results, a CRF model is built on the potential character locations to integrate identification scores, spatial constraints and language model. These components define a cost function whose minimization learning leads to the final word recognition. Figures 2.11(b) illustrates this whole recognition schema.

An almost similar paradigm has been already used by Novikova et al. [NBKL12] for scene word recognition where character candidates are first identified and recognition results are used to build a Weighted Finite-State Transducer (WFST) [MPR02]. The latter presents a unified probabilistic framework that performs maximum a posteriori inference based on character recognition results and linguistic information.

Although the previously mentioned methods perform recognition without prior segmentation of the text image, they are still based on a prior modeling or knowledge about characters in order to infer the whole transcription. This is a part of the temporal classification or the statistical modeling used in HMMs, CRFs or graph models [SH94, Gra12b]. Other segmentation-free methods propose to learn both local and global transcription using a unique framework which is based on RNNs. The network takes as input the whole text image as a sequence of visual features and learns directly the sequence labeling. The difference here is that even the recognition of elementary text units like characters is learned within the whole connectionist process in parallel with the text

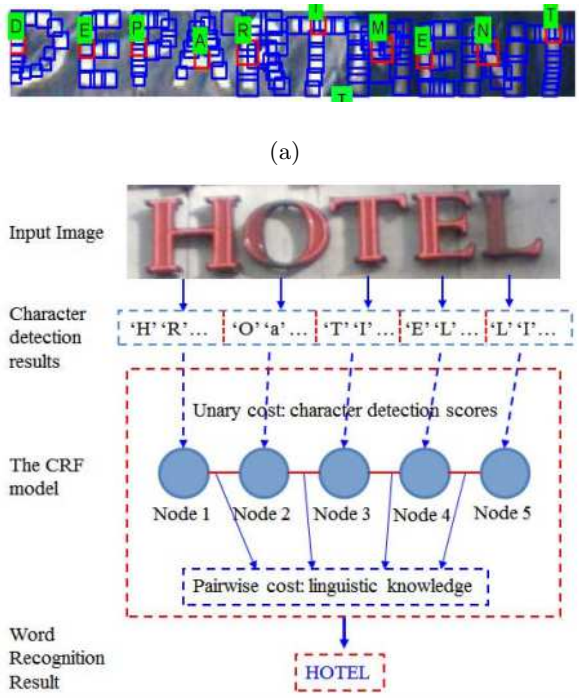


FIGURE 2.11: Illustration of the proposed word recognition method in [SWX+13]. (a) presents an example of the character recognition and detection using tree-structured models. (b) summarizes the whole used word recognition paradigm.

recognition. Another advantage of RNNs is their ability to take into account the context to estimate the probability of each observation. This point has received a particular attention in Bidirectional Long-Short Term Memory-based RNNs (BLSTM). Long range inter-dependencies, in both past and future directions, are considered for sequence labeling. Combined with the Connectionist Temporal Classification component (CTC), the BLSTM can learn text transcription using un-segmented data. This BLSTM-CTC scheme has shown superior performance over HMMs in Arabic and Latin handwriting recognition [GLF+09, Gra12a]. It has also been successfully used for Latin video OCR [EGMS14] with learned features. RNNs have been also used for scene text recognition using a sequential representation of text region with HOG features [SL14].

End-to-end text recognition

Recent advances in pattern recognition methods and also in computation and memory capabilities lead to other text recognition architectures that do not require any explicit or implicit separation between detection and recognition steps. These methods take the whole input image and output the localized and recognized texts. For a relatively small lexicon, the end-to-end recognition seems trivial and can be related to a word spotting process where words are directly identified in the whole image [WB10]. Such methods rely on a holistic paradigm of word images. In [JSVZ16], a ConvNet is trained to classify a large set of word images across a large dictionary. In [RSGP15], both images and labels are projected to the same subspace, learned using a structural SVM, in order to achieve the best matching. For a large or open lexicon, these holistic methodologies are not suitable. Such large word search space requires instead character identification with a

strong character detector and recognizer and sophisticated optimization methodologies to face both the whole image and lexicon complexities.

In [WBB11], an end-to-end text recognition in scene images with a given lexicon has been proposed and has outperformed stepwise recognition where text detection and recognition are separated. The method consists, first, in detecting characters in the whole scene image using a multi-scale sliding window and a Random Ferns-based classifier taking HOG features as input. Word identification is then performed using Pictorial Structures framework that treats characters as 'parts' of a word. For more accuracy, detected words are re-scored using global layout-based features. Non-valid results are suppressed using simple greedy heuristics (non-maximal suppression (NMS) over words). Figure 2.12 illustrates the different steps of this framework. Although the system provides good recognition results, it still depends on a very accurate character recognizer that can distinguish one character not only from other characters but also from all other scene patterns. Moreover, using sliding window to detect characters in a scene image requires special optimization operations to control computational complexities. We note also that the method can handle only words that are within a given lexicon.

FIGURE 2.12: End-to-end text recognition framework used in [WBB11].

Neumann and Matas have proposed, however, two end-to-end system for unconstrained text recognition in real world images [NM10, NM13b]. The first system, proposed in 2010, extracts the character candidates via MSER and filters text candidates using a binary SVM classifier. Another SVM classifier is learned to recognize characters. The system is strongly based on a hypotheses-verification framework simultaneously processing multiple text line hypotheses and strongly uses geometric heuristics and synthetic data for training. Later, in [NM13b], the same authors introduced another end-to-end system that uses oriented stroke features to represent characters, convolving the image gradient field with a set of oriented bar filters. Strokes with specific orientations in a relative position are used to detect and recognize character candidates. The optimization of the recognition results is then performed using Dynamic Programming. Wang et al. [WWCN12] applied the same paradigm based on character identification in order to build and end-to-end text recognizer. The system strongly relies on ConvNet classifiers and is already described in Section 2.2.2.

As presented, end-to-end methods begin to be employed in many text recognition systems. However, regarding applications on video contents, they are still restricted to text in images where a text line is usually limited to one or few words or where the lexicon can be defined. Moreover, no specification has been clearly reported concerning response time.

2.5 Language modeling

Language modeling is a major component in the existing top OCR systems. For example, the OCRopus 0.4 recognizer [Bre08] integrates a MLP for character classification and Dynamic Programming to find the best path in a graph of hypotheses. The language model (LM) consists in a Weighted Finite State Transducer (WFST) that can be seen as a generalization of HMMs and fine state transducers. The used WFST is composed of different models such as character-level and word-level n-grams, stochastic grammars and dictionaries. This representation of the LM as a separate WFST allows OCRopus [BUHAAAS13] to handle different types of documents and languages.

For the Tesseract OCR engine, Smith et al. [LS12] proposed a correction scheme that includes two parallel paths: a LM-based correction path and a document-specific image-based one. The methods have been applied to improve a scanned books OCR that uses basically a dictionary. The image-based component identifies inconsistency between shapes and labels generated by the basic OCR. The used LM encapsulates two parts. The first part is a base LM that consists of word-level n-grams with back-off to character n-grams for out of vocabulary terms. The second part is an adaptive cache model constructed from high confidence words and words verified by the image model and the base LM. It serves for a fair penalization of frequent topic-specific words that appear in a book.

In general, the most frequently used LMs are based on the n-gram statistics to improve recognition accuracy for handwriting or printed texts, in scanned or multimedia documents. In [THR+13], the authors proposed a 5-gram Arabic word-level LM combined with semantically motivated features to improve Arabic handwriting text recognition (HTR). The assembly is used to rerank the n-best hypotheses produced by the Byblos OCR system that relies on HMM decoder. In [WYL09], both character and word-level n-gram LMs are integrated in Chinese HTR. The recognition method is based on character and word segmentation. To find the recognition path, a pruned dynamic programming search is used. It relies on both character recognition and LMs scores.

Elagouni et al. [EGMS14] used a 3-gram LM to enhance two methods of Latin text recognition in videos. The first method consists in segmenting texts into individual characters. The second one is based on a multi-scale scanning window and a graph model to combine window classification results. The LM is used to estimate the joint probabilities of the possible word and paths propositions in the segmentation-based and segmentation-free methods respectively. Hence, the linguistic information can correct some errors related to character confusions or incorrect segmentation. Bissacco et al. [BCNN13] have almost followed the same idea for scene text recognition with a segmentation-based paradigm. They used the best first search algorithm through the graph of segmentation points. The goal is to maximize a score which combines the character classifier and LM likelihoods. For language modeling, they have considered character-level 8-gram LM, word-level 4-gram LM and a dictionary of 100k words.

In [MAJ12b], a framework for scene text recognition has been also proposed. As presented previously, it is based on bottom-up cues derived from individual character detections from the image. A Conditional Random Field (CRF) model has been used to determine the words presented jointly by the detections. At this level, the authors introduced linguistic information as top-down cues. The LM consisted in a strong bigram built from a small lexicon to remove ambiguities between characters and to generate

lexicon-based priors for words. In [MAJ12a], the same authors proposed a framework that uses a large English dictionary instead of an image specific word list. They combined n-gram priors into the CRF model defined on the image in order to recognize scene words which not necessarily belongs to the dictionary. Integrating linguistic information using CRF model has been also used by Shi et al. [SWX+13] for scene text recognition (cf. Section 2.4.3). The proposed LM consists in a bi-gram learned on a large English dictionary with around 0.5 million words provided by [MAJ12a]. The LM priors are incorporated in the pairwise cost function of the CRF model.

Back-off n-grams have been considered as state-of-the-art LM for many years. However, it was always obvious that these frequency-based models are not representative enough of the language modalities. The most important drawback is their inefficiency to represent long-context patterns. The number of possible n-grams increases exponentially with the length of the context. Thus, for massive amount of training data, a large part of patterns cannot be effectively represented and discovered during training by n-grams. Many other advanced LM techniques have been proposed to tackle this problem like Cache LMs [JMRS], Decision Trees and Random Forest (DTRF) LMs [XJ07] and Maximum Entropy (ME) LMs [CMR+09].

Other works proposed to take advantage from feedforward neural networks (NN) in order to learn language modeling. NNs have been successfully used for dimensionality reduction and clustering which is important for unconstrained language modeling in large-scale corpora. The idea behind is to project the context into the same lower dimensional space rather than presenting it as exact sequence of n-1 words. Thus, while learning the prediction of the next word in a sentence, the NN-based LM learns an appropriate real-valued representation of each word. This distinguishes it from the previously mentioned DTRF and ME LMs which require manually designed features before training. NN-based LMs have been introduced more than a decade ago by Elman [Elm90] and Bengio [BDVJ03]. These LMs have been successfully used for automatic speech recognition [SG02] as well as for offline HTR [ZMFEB+14]. However the main drawback of these models remains in their high computational complexity. This explains a large part of researchers' recourse to back-off n-grams since probabilities are generally stored in precomputed tables. Moreover, for feedforward NNs, the context is still limited to several previous words.

Other neural architectures, namely RNNs, have been proposed to address this problem. The effectiveness of the RNN-based LM lies in its representation of the history. Unlike the previously mentioned models, context patterns are learned from data. The history is presented recurrently by the hidden layer of the network and hence not limited to a fixed range. Therefore, the RNN language modelers can handle arbitrarily long contexts. Many variations of RNNs have been proposed to specialize their structure towards effective LMs [KMKB11, MDK+11, MKB+10]. They focus mainly on the application of RNN LMs in speech recognition and machine translation and proved state-of-the-art results compared to feedforward NNs and n-grams.

Later, an interesting work of Graves [Gra13] for automatic text generation based on LSTM networks has been proposed. The idea was to take advantage from LSTM cells to handle long range dependencies in the data. The obtained results compare favorably with those of Tomas Mikolov [Mik12] on Penn Treebank Corpus often used in speech recognition.

2.6 Conclusion

An OCR system can incorporate different operational components including a large panel of pre-processing, detection, localization, recognition and post-processing operations. These parts interact whith each other in different manners which gives rise to different OCR schemes. In this chapter, we have reviewed main approaches and schemes used in this field. Given the similarities between video contents and natural images in terms of challenges, in one hand, and between video and photo OCR methods, in the other hand, we have not limited ourselves on presenting techniques used for videos but we have also reviewed existing methods for both contents. Essential steps in any text recognition cycle, including detection and recognition, were presented and discussed. Moreover, given the impact of language modeling in existing OCR systems' results and recent advances in this filed, a special focus has been given to this component. We presented main language models used in text recognition and their integration paradigm within the whole schema. For Arabic text, existing methods are almost restricted to scanned documents which alleviate their diversity. Therefore, in this chapter, we presented the main advances in this context that are strongly related to the text challenges and not to the document processing.

A first important remarkable point in this overall review is that almost all existing methods that tackle video and image OCR focus on Latin text. This can be related to the focus on English as a global language and also to the lack of multimedia text datasets for other languages and specially for Arabic. Existing OCR products like Tesseract and ABBYY tend to specialize their systems in many languages. However, for Arabic, the performances are still very low compared at least to Latin. A second point that we highlighted is related to the wide use of pre-processing and hand-designed features and priors in most existing approaches. These common features can limit the generalization ability of the proposed method and make it limited to a certain type of data and challenges. A third point that we mention is the complete lack of public Arabic text datasets issued from videos or natural images which represents a very serious challenge for building an OCR system in this context.

To tackle the problem of Arabic embedded text detection and recognition in videos, we propose in our work a specialized framework for this language that is based mainly and completely on data. The specialization aspect aims at providing a sophisticated system for the demanded task instead of targeting the multi-language for generalization purposes, especially with the Arabic script. Besides, we propose to avoid hand-designed priors to the maximum extent at each step of our system. Thanks to machine learning based-strategies, our methods perform text detection and recognition directly on video frames without any pre-processing and are robust to very poor acquisition conditions. All optical and linguistic priors that we are conscious about or maybe others that we do not see are learned from data. In addition, given the lack of available Arabic datasets, we propose in this work new detection and recognition-based datasets issued from Arabic videos, making data and learning techniques and strategies our first concern allows performing best results.

In the following chapters, we will describe our detection, recognition and language modeling approaches. But first of all, we present in the next chapter the experimental infrastructure of our work including the proposed datasets and the different used evaluation metrics.

Chapter 3

Datasets and experimental settings

Contents

3.1 Introduction

As in many computer vision fields, data are the backbone of the OCR systems development. However, for many decades, emphasis has been mainly given to scanned documents. Embedded text on videos and even on photos has received low attention. Therefore, only few datasets have been proposed for this purpose. Moreover, the most important datasets in terms of size and diversity are restricted to some languages, specially to Latin. For the Arabic script, the lack is more remarkable. This has been a very serious difficulty at the beginning of our work. To the best of our knowledge, existing public datasets for Arabic text detection and recognition are limited to handwritten or printed scanned documents or synthetic texts. There is no dataset for Arabic OCR in real-world images or videos. As a solution, we decided to build our own datasets. This work costed a lot in time and effort but it allowed us to provide to the OCR community the first public dataset for Arabic text recognition in videos called *ALIF*.

In this chapter, first, we give an overview of the Arabic script. This is very essential to understand the description of the datasets. Second, we present our proposed datasets for Arabic embedded text detection and recognition in videos, namely the *Detect-set*

and the publicly available *ALIF Dataset*. Finally, we specify the evaluation protocol for both of our text detection and recognition solutions.

3.2 Key features of the Arabic script

The Arabic script consists basically of 28 letters. It is semi-cursive and written from right to left. Letters in a same word are in general connected by a baseline to their predecessor and successor (cf. Figure 3.1). These letters are called joining letters as defined by the Unicode standard. However, there are six letters that accept only to be linked to their preceding letter which creates *paws* in the word (cf. Figure 3.1). For example, the letter 'Hamza' is not a joining letter (cf. Figure 3.1).

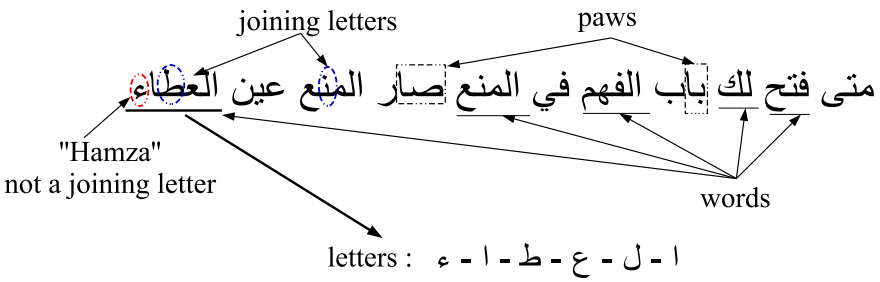


FIGURE 3.1: Cursiveness in Arabic script.

The shape of an Arabic character can change according to its position in the word (isolated, beginning, middle or final). Table 3.1 shows examples of letters and their corresponding shapes (glyphs). In general, letters can have from one to four glyphs. In addition, Table 3.1 shows the high variability in widths between different letters (e.g. letters 'Alif' and 'Baa'). The Arabic script contains also a specific letter that has no semantic sense and no mining in the language. This letter, known as 'Kashida', is a type of justification (expanding baseline between letters in order to adjust left and right alignments).

TABLE 3.1: Examples of Arabic letters and their shapes in different positions.

Letters	Position			
	Isolated	Initial	Middle	Final
Alif	ا	-	-	ـا
Baa	ب	بـ	ـبـ	ـب
Haa	ه	هـ	ـهـ	ـه
Thaal	ذ	-	-	ـذ
TaaaClosed	ة	-	-	ـة

The Arabic alphabet is characterized also by the frequent presence of dots. This alphabet is composed of several groups of letters called joining groups. Each group contains letters having similar shapes but that differ in the position and the number of dots. Some letters have dots above the baseline and others below it. The number of dots varies from one

TABLE 3.2: Joining groups in the Arabic script.

Label	Joining group	Letters
Alif	ا	ا أ آ إ
Baa	ب	ب ت ث
Haaa	ح	ح خ ج
Daal	د	د ذ
Raa	د	د ذ
Siin	س	س ش
Saad	ص	ص ض
Thaaa	ط	ط ظ
Ayn	ع	ع غ
Faa	ف	ف
Gaaf	ق	ق
Kaaf	ك	ك
Laam	ل	ل
Miim	م	م
Nuun	ن	ن
Hamza	ء	ء
Haa	ه	ه ة
Waaw	و	و ؤ
AlifBroken	ى	ى ئ ي

to three dots. Adding a single dot completely changes the letter. Table 3.2 illustrates the different joining groups in the Arabic script.

Arabic script includes also the concept of ligature. Some letters or glyphs when connected together produce another different glyph in a way that they cannot be separable by a simple baseline. An example of ligature between letters 'Laam' and 'Alif' is illustrated by Figure 3.2.

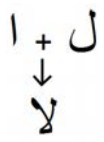


FIGURE 3.2: Ligatures in Arabic script.

These caracteristics of the Arabic script, among others presented in details in [Alg13], make its detection and specially its recognition very challenging.

3.3 Text detection dataset: *Detect-Set*

Our database has been created from three different Arabic TV channels: Al Jazeera, Al Arabiya and France24 Arab. Figure 3.3 presents examples of the used video frames.

FIGURE 3.3: Examples of video frames.

Collected video frames have been manually annotated. We developed a special toolbox for this purpose. The user has to select the text region for each image and an ground-truth XML file is generated automatically. The file contains the name of the image and the localizations of its different text regions (cf. Figure 3.4).

The resulting text regions have been used to create positive examples of training and test sets for the text/non-text classification (cf. Chapitre 4). Negative examples are extracted from a set of images that do not contain text. The training set, denoted **TrainDet** is composed of 30,000 positive examples and initially 20,000 negative examples. Given that we have used bootstrapping in the training procedure (cf. Chapitre 4), we reached at the end 111,000 negative training examples. We used for this 169 large scene images containing different textures. Figures 3.5(a) and 3.5(b) illustrate examples of the used text and non-text patches respectively. A test set denoted **TestDet** has been created in the same way. It is composed of 8,000 text patches and 20,000 non-text patches. It has been used to evaluate the classification rate of our classifiers.

Another two sets of images have also been collected and annotated:

1. **ES1:** A set of 201 frames from Al Arabiya, Al Jazeera and France24 Arabic with 959 texts.
2. **ES2:** A set of 164 frames collected from the BBC Arabic channel. This channel has not been used during training.


```
<imagename>2013-02-16-01-56-10.jpg</imagename>
<resolution width="768" height="576" nbCap="9" />
- <taggedRectangles>
    <taggedRectangle x="157" y="117" width="106" height="50" />
    <taggedRectangle x="150" y="254" width="136" height="66" />
    <taggedRectangle x="304" y="382" width="64" height="27" />
    <taggedRectangle x="473" y="381" width="73" height="28" />
    <taggedRectangle x="322" y="120" width="116" height="48" />
    <taggedRectangle x="319" y="252" width="131" height="71" />
    <taggedRectangle x="505" y="122" width="87" height="41" />
    <taggedRectangle x="543" y="69" width="116" height="42" />
    <taggedRectangle x="497" y="261" width="111" height="59" />
  </taggedRectangles>
</image>
```

FIGURE 3.4: Example of detection-based annotation.

(a) Text patches.

(b) Non-text patches.

FIGURE 3.5: Some patterns used for training: (a) positive examples and (b) negative examples.

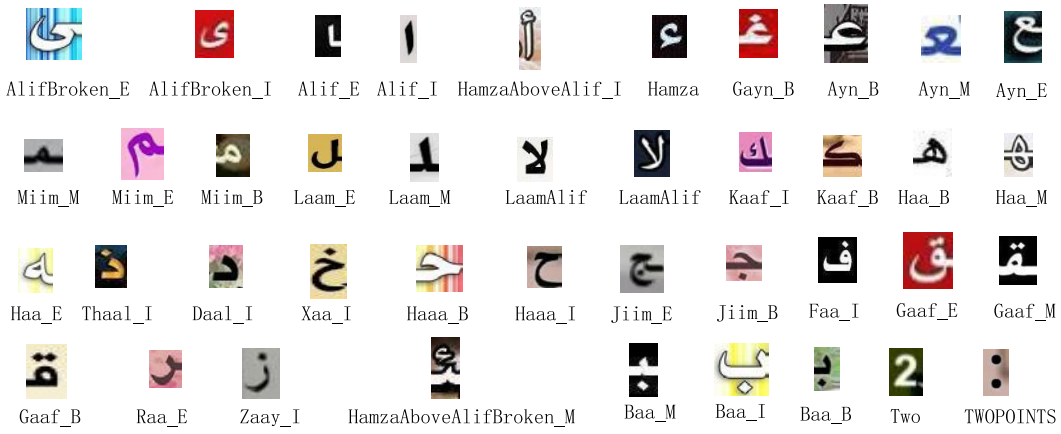
ES1 and ES2 have been used for the evaluation of the final detection results of our methods. In particular, ES2 is used in order to assess the generalization ability of our methods.

3.4 Text recognition datasets

For the recognition scheme, we have constructed two main datasets: the *ArabCharSet* dataset for single character recognition and the *ALIF* dataset [YBG15a] which is composed of text images. We used for this a set of videos from Arabic TV channels of one hour each and some images from the web.

3.4.1 The character dataset: *ArabCharSet*

The *ArabCharSet* has been constructed using 16 hours of Arabic TV broadcast and a small part has been collected from the web. We have manually annotated the appearing texts in terms of character boundaries. We extract, initially from these images, 20,571 character images to which we apply some scaling, noise, color inversion operations. We obtain in total a set of 46,689 images including single letters, punctuation marks and a 'Rubish' class. The different Arabic letters and their possible forms (beginning, middle, end and isolated) are equally represented in this dataset. The ligatures, like the 'لا' case, have been considered as single characters. The 'Rubish' class includes images of non-valid characters (images of multiple characters or parts of characters or parts of the baseline). Examples of character and the 'Rubish' class images are illustrated in Figure 3.6(a) and Figure 3.6(b) respectively.



(a) Examples of character images in *ArabCharSet*.

(b) Examples of the 'Rubish' class images in *ArabCharSet*.

FIGURE 3.6: Some patterns of the *ArabCharSet*. The suffixes '_I', '_B, '_M' and '_E' in the letters names refer to the isolated, beginning, middle and end positions respectively.

3.4.2 The text dataset: *ALIF* dataset

The *ALIF* dataset is composed of **6,532** real Arabic text color images. The majority of these text images has been extracted from five Arabic TV channels: *Al Jazeera*, *Al Arabiya*, *France 24 Arabic*, *BBC Arabic* and *Al Hiwar*. We have used in total 64 videos, of one hour each, recorded in different periods between 2013 and 2014. Extracted images from these videos include people and places names, events, news, dates, speech translation, etc. The rest, nearly 12% of the text images, has been collected from *Facebook* (*FB*) pages of other Arabic TV channels like *MBC*, *MBC Masr* and *Al Kods*. The number of images and the corresponding sources are summarized in Table 3.3.

TABLE 3.3: The source and the number of collected text images.

Source	# of text images
Al Ajazeera	3257
Al Arabiya	675
France 24 Arabic	1605
BBC Arabic + Al Hiwar	225
FB images	770

The collected text images represent Arabic embedded text in TV broadcast frames. Different fonts (more than 20), sizes (varying from 14*bp* to 260*bp*), colors, backgrounds, luminosities, contrasts and occlusions are represented in the dataset. Examples of collected text images are presented in Figure 3.7.

FIGURE 3.7: Diversity of text images in the dataset.

The dataset contains in total **89,819** characters, **52,410** paws and **18,041** words. During the manual annotation, we have considered 140 Arabic character glyphs or forms:

- 122 letter forms including 'white space' (we take into account diffrent positions of each letter),
- 10 Arabic digits,
- 8 punctuation marks.

Table 3.4 gives for each character, its frequency in the dataset.

TABLE 3.4: Distribution of letters in the dataset.

Character	# of times	In Arabic
Alif	13209	ا
Taaa	3860	ت
Jiim	1385	ج
Xaa	683	خ
Thaal	232	ذ
Zaay	693	ز
Shiin	1007	ش
Daad	472	ض
Taa	211	ظ
Ghayn	480	غ
Gaaf	1968	ق
Laam	8263	ل
Nuun	4557	ن
Waaw	4651	و
Hamza	310	ء
Haa	1267	ه
Zero	196	0
One	134	1
Two	188	2
Three	104	3
Four	71	4
Five	60	5
Six	44	6
Seven	50	7
Eight	48	8
Nine	42	9
DASH	90	-
COMMA	21	،
POINT	206	.
SLASH	16	/

Character	# of times	In Arabic
Baa	3165	ب
Thaa	419	ث
Haaa	1866	ح
Daal	2823	د
Raa	5518	ر
Siin	2971	س
Saad	911	ص
Thaaa	964	ط
Ayn	2606	ع
Faa	2277	ف
Kaaf	1529	ك
Miim	5130	م
Yaa	7895	ي
HamzaAboveAlifBroken	451	ئ
TildAboveAlif	52	آ
HamzaUnderAlif	424	إ
Laam_HamzaAboveAlif	480	لأ
Laam_TildAboveAlif	24	لآ
TaaaClosed	2985	ة
HamzaAboveWaaw	113	ؤ
AlifBroken	559	ى
HamzaAboveAlif	860	أ
Laam_Alif	880	لا
Laam_HamzaUnderAlif	174	لإ
INTERPOINT	57	؟
EXCLPOINT	12	!
TWOPOINTS	143	:
PERCENTAGE	17	%

3.4.2.1 Dataset annotation

The dataset has been manually annotated. We have developed our own interactive application for this purpose. The user of the application selects the text in the video frame and enters the corresponding transcription. The application automatically generates the metadata file and the cropped text images. Metadata are provided using the *XML* format. For each text image, two different kind of metadata are provided: (1) global information that includes the text image name and its size, and (2) the text transcription given with Arabic characters and Latin labels.

As previously mentioned, almost 35% of the text images in the *ALIF* dataset is finely annotated. This subset is denoted $ALIF_{fine}$. For this subset, boundaries and transcription of each word, each paw and each character in the text image have been manually annotated. Hence, these fine additional information are incorporated in the metadata file of each text image in $ALIF_{fine}$ as an extension of the annotation previously mentioned.

(a) For a text image from *ALIF*.

(b) For a text image from $ALIF_{fine}$.

FIGURE 3.8: *ALIF* and $ALIF_{fine}$ datasets annotations.

Figure 3.8 illustrates two examples of metadata files: one for a text image from *ALIF* and another one for a text image from the $ALIF_{fine}$. The main differences between the *ALIF* dataset and its subset $ALIF_{fine}$ are summarized in Table 3.5.

3.4.2.2 Dataset organization

As mentioned in the introduction, the *ALIF* dataset is mainly dedicated to train and evaluate techniques for Arabic embedded text recognition in videos. In order to make *comparable* different studies that will use this dataset, we propose to partition it into

TABLE 3.5: *ALIF* and $ALIF_{fine}$ metadata.

	# of images	**Metadata**
ALIF	6,532 (including $ALIF_{fine}$)	- image name, width and height - whole text transcription in Arabic and Latin labeling
$ALIF_{fine}$	2,308	- image name, width and height - whole text transcription in Arabic and Latin labeling - words, paws and characters boundaries in the text image - words, paws and characters transcriptions in Arabic and Latin labeling

training and testing subsets. Those who will use our training dataset and provide results on it can make comparable the effectiveness of their learning techniques. If *ALIF* is used for testing, performances can be reported per subset, such that different techniques can be directly and objectively compared.

The subsets we propose are the following. The proposed subsets for testing are obviously disjoint from the train dataset.

1. ***ALIF_Train***: is composed of **4,152** text images. It is dedicated for training. The images are selected in a way to cover a large range of variability in texts and acquisition conditions.

2. ***ALIF_Test1***: is composed of **900** text images that have been collected from the same sources as ***ALIF_Train***.

3. ***ALIF_Test2***: is composed of **1,299** text images. It has the same characteristics as ***ALIF_Test1*** but provide about 400 additional text images.

4. ***ALIF_Test3***: is composed of **1,022** text images. It has been collected from sources that has not been used at all during the construction of the previous subsets. It is therefore very different from the train dataset and can be used to assess the generalization capabilities of techniques that has been trained using ***ALIF_Train***, in very challenging conditions.

The *ALIF* dataset has been created in an incremental manner. It can be obtained upon request. Instructions are explained in the following webpage: https://cactus.orange-labs.fr/ALIF/.

3.5 Evaluation protocol

3.5.1 Metrics for the text detection

The evaluation of the final detection results is not a trivial task. It is almost impossible for a detected text region to be exactly the same as expected detection (the ground-truth). In addition, the lack of common test databases and ground-truth is a real obstacle towards an objective comparison between existing text detection methods. Researchers often use simple metrics for the detection evaluation: boxes-based and pixel-based recall and precision values. In text detection, split and merge cases are frequent. One ground-truth can correspond to more than one detection and vice-versa.

In our work, we have used the metric proposed in [WJ06] to evaluate our methods. It is based on a matching graph between the ground-truth and the detection results where, for one ground-truth box G_i and one detection D_i, recall and precision values are expressed as follow:

$$\begin{aligned} r_{ij} &= Area(G_i \cap D_j)/Area(G_i) \\ p_{ij} &= Area(G_i \cap D_j)/Area(D_i) \end{aligned} \tag{3.1}$$

Three types of matches are considered:

- One ground-truth and one detection match if $r_{ij} > s_r$ and $p_{ij} > s_p$, where s_r and s_p are two thresholds for recall and precision respectively.
- Split case (one-to-many matches): one ground-truth matches against a set of detections if those detections cover a large portion (greater than s_r) of it and each of them overlaps enough with this ground truth (overlap greater than s_p).
- Merge case (many-to-one matches): one detection matches against a set of ground-truth. This is the inverse case of split.

Considering these cases, the precision and recall metrics are expressed as follow:

$$\begin{aligned} R(G, D, s_r, s_p) &= \textstyle\sum_i \mu(G_i, D, s_r, s_p) / |G| \\ P(G, D, s_r, s_p) &= \textstyle\sum_i \mu(G, D_i, s_r, s_p) / |D| \end{aligned} \tag{3.2}$$

where $\mu()$ is the match function. It is evaluated to 1 in case of one-to-one match, to 0 if there is no match and to 0.8 in case of split or merge. This last value is a penalization of one-to-many and many-to-one matches. In our evaluation, recall and precision thresholds s_r and s_p have been set to 0.6 and 0.3 respectively.

In addition to the recall and precision metrics, we report also the detection rate (DR), the number of false alarms (#FA) and the response time.

3.5.2 Text recognition metrics

Given that our recognition methods are segmentation-free, we measure their performances based on the Edit Distance between the ground-truth and the recognized text.

Therefore, we use the following metrics: the character recognition rate *(CRR)*, the word recognition rate *(WRR)* and the whole text recognition rate *(TRR)*.

During evaluations, we remarked that the *WRR* does not fairly reflect the performance of our methods. Indeed, a large number of words are considered as not recognized often because of one wrong character (cf. Chapitre 5). Consequently, to take this finding into consideration, we added an additional evaluation metric: the word recognition rate including words with at most one wrongly recognized character *(WRR-1C)*.

Thus, given the recognized text (RT) and the ground-truth (GT), these measures are computed as follows:

$$CRR = \frac{\#characters - \sum EditDistance(RT, GT)}{\#characters}$$
$$WRR = \frac{\#words\ correctly\ recognized}{\#words}$$
$$WRR - 1C = \frac{\#words\ with\ at\ most\ one\ wrong\ character}{\#words}$$
$$TRR = \frac{\#text\ images\ correctly\ recognized}{\#text\ images}$$

3.6 Conclusion

In this chapter, we have presented the main primary tools to build and evaluate our text detection and recognition methods. As there is no publically available Arabic text datasets issued from videos or natural images, a special focus was given in this chapter to the built datasets. The used text and character images include a wide diversity in terms of text specificity and acquisition conditions. This reflects a part of the tasks difficulty. This chapter was dedicated also to present the specificity of the Arabic script. The goal behind is to facilitate the comprehension of the datasets structure and to illustrate the additional challenges related to the nature of the script.

The next chapter is dedicated to the text detection. The different methods proposed for this task are presented and evaluated on the proposed dataset with respect to the mentioned detection metrics.

Chapter 4

Arabic text detection in videos

Contents

4.1 Introduction

Unlike scene text, captions or embedded text in videos is not necessarily limited to few words. It can consist in one or multiple lines with different lengths. Therefore, detecting embedded text lines in videos is a very essential step before recognition. Detection consists in separating text regions from the background and precisely localizing it in the video frame using bounding boxes for example. The task is straightforward with uniform background where a simple image[1] thresholding can resolve the problem. However, in the case of video contents with cluttered background and complex environment (varying colors, contrast, luminosity, etc.), text detection becomes a challenging task. This is related, in a first level, to the complexity of the discrimination between text and non-text regions and to the difficulty in reaching precise localization. Particularly, for the Arabic text, many additional text challenges are faced. The Arabic text has different texture characteristics compared to Latin or Chinese ones: more strokes in different directions, different fonts and character aspect ratios, more diacritics above and below characters, etc.

In our work, our text detection methodology addresses the Arabic script in particular and considers the task as a pipeline of two sub-tasks: image region classification and

[1]The word 'image' refers to a video frame or an image in general.

text localization. The general approach takes raw pixels as input and do not rely on any preprocessing of the video frames. The idea is to build a strong text/non-text classifier with high discrimination ability between Arabic text and the background. The application of such a classifier on the input image defines a set of text region candidates or connected components. The localization step consists in applying a grouping algorithm to these components in order to form text lines with precise definition of their positions relative to the image and their dimensions in terms of width and height i.e the position and the dimensions of the bounding box that limits precisely the text.

In this work, we propose three machine learning-based methods for Arabic text detection [YBG14]. The first one is based on a Convolutional Neural Network (ConvNet). It performs both text feature extraction and classification. The second detector is based on hand-crafted features, namely Multi-Block Local Binary Patterns (MBLBP). Text/non-text region classification is learned using a particular Boosting cascade where relevant MBLBP features are selected using the Gentleboost algorithm. We propose to compare this method to a third detector that relies almost on the same Boosting schema but using instead the common combination of Haar-like features and Adaboost algorihm. A robust localization of Arabic text lines is then performed without applying any tedious geometric constraints or local image pre-processing.

This chapter presents the proposed text detection approaches. We describe, first, the proposed neural and boosting-based classifiers in Section 4.2, then, the text localization procedure in Section 4.3. Experiments and results are presented and discussed in Section 4.4. Finally, Section 4.5 provides some conclusions and highlights some of the advantages and drawbacks of each of the proposed methods.

4.2 Text regions classification

The first step in our text detection procedure is to build a strong classifier that can discriminate between text and non-text patterns. It is not a trivial task given its high level of asymmetry. It consists in finding a discriminative model between the 'text' patterns and all the other patterns that may exist in an image or video frame which are unconstrained, corresponding to the rest of the world patterns. ConvNets and Boosting approaches proved to be among the most powerful tools used for pattern classification including faces, objects and texts [VJ01a, VJ04, LB95, LBBH98, DG08, GD04]. In the following sections, we review the two methods in details while depicting their characteristics and differences between them. We give also our training architectures and procedures used to build our Arabic text/non-text classifiers.

4.2.1 Convolution Neural Network-based detection

Convolution Neural Networks (ConvNets) are special class of feedforward Artificial Neural Networks that allow to learn the extraction and the classification of visual features directly from data using one holistic neural architecture. In this section we give an overview of the origins of this type of networks and its features. Then we describe the ConvNet architecture that we use for the Arabic text detection and and the training process.

4.2.1.1 An overview of Convolution Neural Networks

In general, one of the main advantages of neural networks, namely Multi-Layer Perceptrons (MLPs) [RHW85], is their ability to learn complex data modalities. Thanks to its hidden layer(s), it can project the input data into a space where it becomes linearly separable. Given a set of training examples and a supervised learning algorithm, they can hence approximate non-linear functions. Usually, used algorithms are based on the gradient descent, like the backpropagation algorithm [Wer88a].

Theoretically, for pattern classification in the field of computer vision, MLPs can learn feature extraction and classification if applied directly on raw pixels of data images. However, the problem is that to do so, they need huge amount of data to converge. For a small input image of 24-by-24 for examples, the network mission is to classify data images of dimension 576 which implies a neural architecture with a large number of parameters. Learning these latter requires thus large number of training images otherwise it over-fits. In addition, for complex data images, this required number becomes much larger in order to explore and learn different data modalities. That is why, MLPs are usually applied on a set of empirical or hand-crafted features extracted beforehand from data. This reduces the input dimensionality avoiding thus the over-fitting and allows the MLP to focus only on the classification task. However, in that case, their efficiency still depends always on how much the chosen features are discriminant.

Convolution Neural Networks [FM82, L+89], are bio-inspired neural models that are proposed to solve this problem using a special architecture that takes into account spacial information and combines three main concepts:

- **local receptive fields** that allow to detect local features.
- **weights sharing** that allows to replicate the search of these features and thus, implies a reduction of the network parameters and avoids over-fitting.
- **spatial sub-sampling** which reduces the sensitivity to some small translation, rotation and scale variations and to low distortions.

First forms of ConvNets have been proposed by Fukushima [Fuk75, FM82] which are inspired from the work of Hubel et Wiesel [HW62] on the cat's visual cortex. The proposed model, called Neocognitron, is a sequence of neuron layers, in which neurons are organized in the form of 'feature maps'. Each neuron in a given feature map is connected to only one neighborhood corresponding to a fixed number of neighbor neurons in one or more feature maps of the previous layer. This neighborhood defines the **local receptive field** and its associated connection weights are learned in a way to extract relevant elementary forms like corners and edges [Fuk75, FM82, LeC86, L+89]. Different feature maps can be defined for a given input image, each one has its own connection weights. However, **weights are shared** by all local receptive fields in a given feature map applied in different locations of the corresponding input maps in the previous layer. Therefore, each feature map in a given layer is the result of a convolution of its input feature map(s) by a mask defined by the corresponding connection weights. This mechanism defines the convolution layers.

The maps of these layers are then **sub-sampled** in order to reduce the sensitivity of the network to small variations of scales, forms or centering in the input image. Similarly,

the connection weights between the two layers are automatically learned. The succession of alternated convolution and sub-sampling layers defines the architecture of the Neocognitron. For a classification task, the output layer corresponds to a classification layer of the extracted hierarchical features. It is defined by a set of neurons, each corresponds to a class. A good classification of an input image corresponds to the activation of the neuron of the corresponding class.

The particularity of such a classification model is that all the parameters (weights) relative to the feature extraction, combination and classification are learned. Hence, the sophistication of the network strongly depends on the used training procedure. First training algorithms were layer-wise [Fuk88]. Layers are trained one by one in a supervised manner. Feature maps are trained to detect features fixed *a priori*. Later, a convolution network has been proposed by Lecun et al. [LDH+90] based on the work of FuKushima [Fuk88]. The most important particularity that makes the network reaching an apogee of success and efficiency is related to the training procedure. Instead of a layer-wise training and instead of fixing some *a priori* features to learn at each map of each layer, the authors proposed an end-to-end training algorithm. The network receives training images and their classes as input and learns to minimize a global error function using the backpropagation of gradients algorithm. This training process allows the network to automatically infer relevant features and to learn the parameters of their combinations and classification that fit the target class. Besides, the proposed network has a lighter architecture then the initial Neocognitron.

Many architectures derived from this proposed ConvNet or CNN have been proposed to fit different problems like document reading, OCR and handwriting recognition [LDH+90, SSP03, LBBH98, SG07a], face recognition [LGTB97, TYRW14], detection of faces, texts, pedestrians and human bodies in natural images [SKCL13, VML94, GD04, OLM07, DG08, TGJ+15], speech recognition with Time-Delay Neural Networks [WHH+89] and traffic sign recognition [CMMS12]. Recently, a review about ConvNets, deep learning architectures in general, their different applications and related works has been given by Lecun, Bengio and Hinton in [LBH15].

4.2.1.2 The used ConvNet classifier

In our work, we use ConvNets in order to build an Arabic text/non-text classifier, inspired by the model proposed in [DG08]. Using a large training set of text and non-text patterns, the network automatically drives feature extractors that best discriminate between the two classes of patterns. The model do not make any assumptions about the features to extract and do not rely on any pre-processing on the input images.

The used ConvNet architecture is illustrated in Figure 4.1. It consists in a pipeline of six layers. It receives training labeled images with a retina of 32×64 pixels. The first four layers perform feature extraction and combination. The two last ones are dedicated to feature classification. Each layer contains feature maps resulting from convolution, sub-sampling or neuron unit activations from previous layers' outputs.

The first layer $C1$ is a convolution layer with $n_{C1} = 5$ feature maps of 28×60 pixels each. Each unit in each feature map is connected to a 5×5 neighborhood of the input retina in a contiguous manner. Each unit corresponds to a convolution by a 5×5 trainable mask, followed by the addition of a trainable bias. Connection weights are shared by

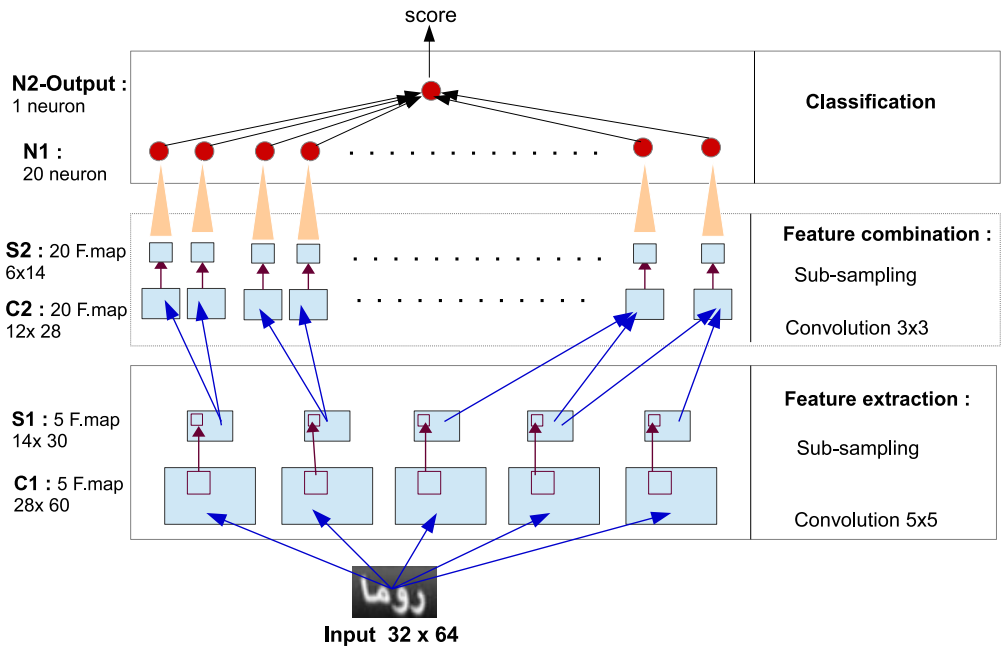


FIGURE 4.1: Convolution Neural Network architecture.

units of each map. The different features maps in this layer correspond to different trainable masks and basis. Each map has, thus, 25 trainable weights plus a trainable bias which leads to 130 (5×26) trainable parameters for this $C1$ layer. Hence, in each location of the input image, different features are extracted i.e. multiple maps lead to the extraction of multiple features (corners, oriented gradients, etc.) directly from the input image pixels.

Each of these maps is then sub-sampled in the second layer $S1$ which reduces by 2 their spatial resolution. This layer contains $n_{S1} = n_{C1}$ feature maps reducing the sensitivity to affine transformations of small amplitudes. The receptive field of each unit in these maps corresponds to a neighborhood of 2×2 in $C1$. The value of the unit corresponds to the average of its neighborhood multiplied by a trainable weight, added to a trainable bias and passed to a non-linear activation function. This layer has thus 10 trainable parameters. Feature maps are of 14×30 pixels each.

Layer $C2$ is a convolution layer. It contains $n_{C2} = (n_{S1} \cdot 2) + n_{S1}! = 20$ maps. The implementation of the convolutions are similar to this in layer $C1$ but with different types of connections. Each map of $S1$ gives two maps by application of convolutions with two different 3×3 weighted masks, resulting in the first 10 feature maps with 100 trainable parameters (9 convolution weights and 1 bias for each of the 10 maps). Each pair of maps of $S1$ are combined in one map after applying a convolution with two different 3×3 kernels which gives the 10 remaining maps with 10 bias and 90 (9×10) trainable mask weights. In total, we have 200 trainable parameters for the $C2$ layer. The size of each feature map is 12×28 pixels. This duplication and combination of the feature extractors lead to the exploration of other high-level features. Layer $S2$ is a sub-sampling layer of $C2$ maps. It contains $n_{S2} = n_{C2}$ feature maps of 6×14 pixels, each with 40 trainable parameters.

The last two layers form a simple MLP that performs feature classification. They are composed of 20 and 2 standard sigmoïd neurons respectively. Each neuron in the layer

$N1$ is fully connected to only one feature map of $S2$. Layer $N2$ contains one neuron indicating the class of the input image (1 for 'text' class and -1 for 'non-text' class). The whole architecture has only 2100 trainable parameters corresponding to some hundred thousands of connections.

4.2.1.3 Training procedure

The network is trained to extract appropriate text image features and classify text and non-text images. For training, we use the subset *TrainDet* of the *Detect-Set* dataset described in Section 3.3 of the previous chapter. The dataset contains 30,000 text patches of an aspect ratio of 2. This ratio is chosen for a better presentation of the Arabic text structures. Indeed, with the presence of the specific Arabic letter 'kashida' that is used sometimes to expand the baseline between letters in order to adjust left and right alignments (c.f Section 3.2), setting the aspect ratio to a small value may cause missing of some Arabic strokes in a patch. The non-text training images have the same size with initially 20,000 examples randomly extracted from natural scene images that do not contain Arabic texts. In order to improve the precision level of the model in localizing Arabic text lines, we incorporate into the negative training set some patches that present badly cut text and multi-line regions.

As we previously presented, the *TrainDet* dataset is divided into two sets: training and validation. At each iteration, the ConvNet is trained with an equal number of positive and negative examples randomly selected from the training set. For each example i, a mean square error $MSE = (o_i - d_i)^2$ is computed, where o_i is the network response and d_i is the desired output. The error is then back-propagated throw all layers to update weights. The validation set, which remains constant at each iteration, is used to check the generalization ability of the network during training and to avoid over-fitting by selecting the configuration that performs best on it.

Bootstrapping
In order to boost the rejection ability of false alarms, we use a bootstrapping technique. After each training epoch containing 50 iterations, a set of false alarms are gathered by running the ConvNet on various large images that do not contain any text. These false alarms are added as negative examples to the training set. At each epoch, a grabbed example is considered as false alarm if the network response is greater than a threshold Thr. We gather at maximum 8,000 false alarms at each epoch. Initially, $Thr = 0.8$ when the network is still a weak classifier. This threshold is then gradually reduced by 0,1 at each epoch as the network becomes more and more sophisticated. At well advanced epochs, the network, evaluated on the validation set, gives very low classification errors. At this level, the gathered false alarms correspond to few patterns that are very hard to discriminate from text patterns (texture very close to the Arabic text). Some examples of negative training images have been presented in the previous chapter (Section 3.2). At the end of the training procedure, we reach a negative training set of almost 80,000 non-text images.

4.2.2 Boosting-based detection

The second proposed detector is based on Boosting to build a binary text/non-text classifier. Unlike the ConvNet classifier, this method consists in two fundamental separated

steps: feature extraction and classification. In an image, pixel values can give information only about luminance or color. It is therefore wiser to find operators that can capture more global features to describe different structures in the image. In our work, we propose to use the Multi-Block Local Binary Patterns (MBLBP). A global Boosting structure is then used to learn classification, namely the multi-exit asymmetric Boosting cascade that learns to distinguish text and non-text using the MBLP features while incorporating the Gentleboost algorithm.

Moreover, we propose to compare this architecture to another one that is based also on the multi-exit asymmetric Boosting cascade but using the well known combination of the Haar features with Adaboost algorithm. Therefore, under the Boosting-based strategy, two detectors are build.

4.2.2.1 Feature extraction

The main idea behind MBLBP features, proposed by Zhang et al. [ZCX+07], is to encode rectangular regions in an image using the Local Binary Pattern (LBP) operator [OPH94, OPH96]. This latter has been proposed to describe local textural patterns and, in its original version, it operates in image blocks of 3×3 pixels. It consists in labeling each pixel of the image by a value that indexes the local pattern of this pixel. For this, the pixel intensities in each block are thresholded by the intensity of the center pixel (of the block) which produces a binary code. The decimal representation of this code is used as a label for the center pixel. In this case, the neighborhood consists in 8 pixels which gives in total $256 = 2^8$ kinds of possible patterns.

For the MBLBP operator, a binary code is produced by comparing the average intensity of the central rectangle r_c with its 3×3 neighborhood $X = \{r_0, \ldots, r_8\}$.

$$MBLBP = \sum_{i=0}^{8} \delta(r_i - r_c) 2^i \tag{4.1}$$

where

$$\delta(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x < 0 \end{cases} \tag{4.2}$$

Figure 4.2 presents an example of MBLBP feature produced in an image region. We note that not all the feature configurations that can be defined by the operator must be used or found in an image.

In our work, for a given text or non-text patch, we consider different sizes of rectangular blocks. Each block is defined by its height, width and its center coordinates in the image. An index is thus given to each block. The MBLBP attributed to each block is a value in the interval $[0, 255]$ given that we consider always a neighborhood of 3×3 blocks. We note that we avoid extreme cases for block patterns; for example, when a central block and its neighborhood cover the whole image. As defined, the MBLBP models are able to capture different structures in the image like edges, corners and flat surfaces at different scales and positions. These features are more diverse than those produced by the original LBP operator that focuses on very elementary structures.

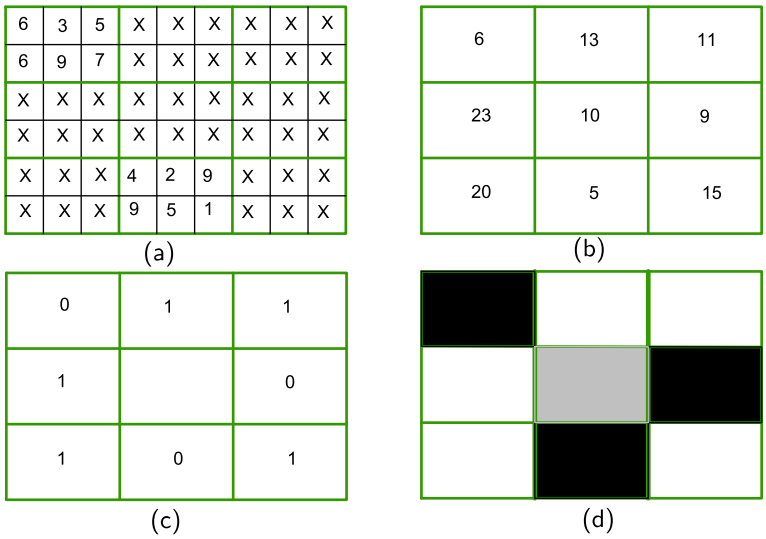


FIGURE 4.2: MBLBP: (a) region of an image divided into rectangles, (b) the average values of rectangles are calculated, (c) comparing central rectangle value with its neighborhood rectangles to provide a binary sequence defining in (d) the MBLBP relative to this region.

In addition to the MBLBP features, we propose also to use the Haar wavelets. Haar-like descriptors have been popularized by Viola and Jones for face detection [VJ01b]. They are based on the difference measure between the average intensities in contiguous rectangular regions. Hence, they encode, contrasts and spatial relations in a given image. Haar features can also capture intensity gradient at different locations and directions by changing the size, the shape and/or the position of rectangular regions.

As presented, both MBLBP and Haar-like features are very simple to implement. However, given the variations in sizes and positions, they become too numerous. Later, in the text localization step, each model will be used in a scanning scheme of large images which will be very time consuming. Thus, in order to improve computational efficiency of these two descriptors, we use the classical *Integral Image* technique [VJ01b].

4.2.2.2 Classification

Boosting-based feature selection

Boosting is a classification method that allows to combine multiple simple or weak hypotheses or features in order to create a stronger hypothesis that has higher discrimination ability between class patterns. In other terms, it consists in boosting a set of weak classifiers in order to build more accurate strong classifier.

Weak classifiers, called also 'rule of thumb' or 'base learners', refer to hypotheses that produce a classifier with high empirical risk. They perform just slightly better than random guessing. For a labeled training set presented as a set of Haar like features for example, the role of a weak classifier is to find a basic rule defined on these features in a manner to minimize the classification error over this set. This can be a threshold-based binary classifier which is created from each Haar feature so that the weighted training error is minimized. Therefore, the Boosting algorithm is used to select relevant

discriminative features. In other terms, the Boosting is used to learn a function f that separates the best between classes. Generally, it is expressed as follow for a given example x and a set of weak hypotheses $h_1, \cdots, h_K$:

$$f(x) = sign(\sum_{k=1}^{K} \alpha_k h_k(x)) \tag{4.3}$$

where α_k are weight contributions of the hypotheses. The definition of the weights and the selection of relevant hypotheses are learned by a training process using a set of labeled data.

In this work, we explore the Gentleboost algorithm applied in a more global cascading structure that we will describe later. It is used to learn the selection of relevant MBLP features. The resulting classifier is compared to another Boosting scheme based on the same cascading structure but incorporating the classical combination of the Adaboost algorithm and the Haar-like features.

Adaboost or Adaptive Boosting [SS99, Sch03] in one of the most used Boosting algorithms. It is called adaptive given that it maintains, over training iterations, a weight distribution over training examples so that weights of badly classified examples should rise and those of well classified examples should decrease. Given a training set $(x_1, \cdots, x_N)$ of N examples, their labels $(y_1, \cdots, y_N)$ and a number K of iterations, the algorithm can be described by three main steps for a binary classification task:

1. Uniformly initialize the weights of the training examples (w_i)
2. For k=1 to K
   - Fit a classifier h_k to the data regarding current weights w_i
   - Compute the error rate of the current classifier ϵ_k
   - Compute the weighted contribution of the classifier $\alpha_k = 0.5\frac{1-\epsilon_k}{\epsilon_k}$
   - Update the weight of each input pattern or example $w_k \leftarrow w_k exp(-\alpha_k I(y_i \neq h_k(x))$
3. Output the weighted combination of classifiers as a final strong classifier (c.f Equation (4.3))

where

$$I(x, y) = \begin{cases} 1, & \text{if } x \neq y \\ 0, & \text{if } x = y \end{cases} \tag{4.4}$$

A simple manner to create a correspondence between weak classifiers and extracted features is to attribute each classifier to a single feature. At each iteration, the single best weak classifier at this iteration is chosen, corresponding to a single feature. The weight confidence of this feature is updated at each iteration based on its classification error on the new weighted training examples issued from the last iteration. By updating the weights of training examples, weak classifiers are forced to focus on the hard examples in the next iteration.

Several variants of Boosting algorithms have been proposed like Realboost [FHT+00], Gentleboost [FSA99, SS99], Reweight Boost [VV05], etc. Usually, changes, regarding

Adaboost algorithm, are related to the definition of weak classifiers and to how their errors and weighted contributions are calculated. In particular, the Gentleboost algorithm, that we use in our work, is a modified version of the Realboost algorithm. This latter gives a new formulation of weak classifiers based on the calculation of the probability that a given pattern or feature belongs to a class. Given this continuous aspect in outputs, it is able to perform an exact optimization of these classifiers. The Adaboost algorithm, however, is based on classifying the input patterns and calculating the weighted amount of error which makes weak classifiers hard to optimize. The Gentleboost algorithm, however, is an improvement of the Realboost algorithm by using weighted least-squares regression for the estimation of weak classifiers. For Realboost, this estimate is based on log-ratio which cause very large update values and makes it numerically unstable. The Gentleboost algorithm proves to be more robust to noisy data and more resistant to outliers.

Multi-exit asymmetric boosting cascade and training procedure
In our work, as we previously mentioned, we propose to integrate the boosting learning mechanism in a more global cascading architecture for further improvement in terms of discrimination and generalization. In order to build the final Arabic text/non-text classifier, we propose to use the multi-exit asymmetric boosting cascade introduced in [PHC08]. In this cascade, intermediate strong classifiers are represented by a set of nodes having indices ℵ. These intermediate classifiers are the result of applying the boosting algorithm (Gentleboost or Adaboost) on the the extracted features (MBLBP or Haar-like features). Each classifier (node) makes a decision to pass or reject the input subwindow. Each strong classifier is constructed from a sequence of n weak classifiers. Unlike conventional cascade proposed in [VJ01b], nodes are able to use overlapped sets of weak classifiers, i.e. each node exploits these weak hypotheses from the beginning to a number n which corresponds to a fixed training false acceptance and false rejection target rates. Thus, this approach needs much less weak classifiers than traditional cascades.

For our training procedure, we use the *TrainDet* dataset composed of Arabic text and non-text patches issued from Arabic videos. During the training, the cascade takes as input text and non-text images presented as hand-crafted features which are considered as weak classifiers and two thresholds α_0 and β_0 for false acceptance and false rejection target rates. As shown in Figure 4.3, at each node, a boosting algorithm (Adaboost or Gentleboost) selects relevant features and train a new relatively strong classifier H_n that rejects negative examples and passes positive examples to the next node. As we previously mentioned, in this cascade, the number n of the weak hypotheses used at each node corresponds to the fixed false acceptance and false rejection target rates α_0 and β_0. This can be achieved by formulating the problem as follow:
If we want our boosted classifier $H_n(x) = \sum_{i=1}^{n} w_i h_i(x)$ to achieve a false acceptance rate $FAR(H_n(x)) \leq \alpha_0$ and a false rejection rate $FRR(H_n(x)) \leq \beta_0$, we have to train our weak classifiers $h_i(x)$ by minimizing the following asymmetric goal:

$$G = \beta_0 FAR(H_n(x)) + \alpha_0 FRR(H_n(x)) \tag{4.5}$$

In other words, a reject in a node is done only if $FAR(H_n(x)) \leq \alpha_0$ and $FRR(H_n(x)) \geq 1 - \beta_0$. The thresholds are fixed in an asymmetric manner reflecting two asymmetry aspects in this detection task. The first asymmetry is related to the unequal distribution of data: rare text regions have to be distinguished from the enormous number of possible

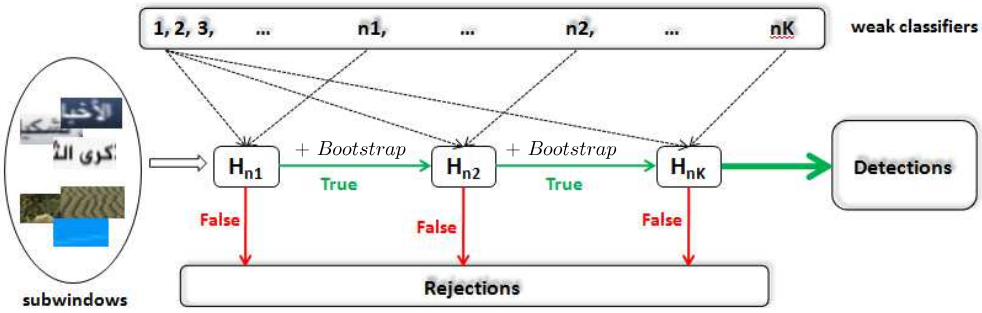


FIGURE 4.3: Multi-exit boosting cascade structure

non-text regions present in video frames. The second asymmetry is related to the difference between the target detection and reject rates during training. On one hand, we aim at reaching very high detection rate because we do not want to miss any Arabic text region. On the other hand, given the high number of non-text regions, a very low number of false alarms is desired for a better reliability. In our work, best training results are achieved with $\alpha_0 = 0.8$ and $\beta_0 = 0.01$

Similarly to the ConvNet classifier, during the training of this boosting cascade, we use the bootstrapping procedure in order to further improve the rejection ability of false alarms. We use for this a set of large images that do not contain Arabic text, the same used by the ConvNet. After each reject point, we apply the last strong classifier on these images in order to gather false alarms, of at maximum 8,000 patches, that we add to the negative training set.

Finally we note one other crucial point about this training procedure. Given that the classification problem becomes more and more difficult through nodes, it is possible that reaching the desired rates becomes very hard or even impossible. Therefore, a stop condition is defined for the training algorithm: the training is stopped when the number of weak classifiers between two nodes exceeds a *a priori* fixed value that depends on the used boosting algorithm (Gentleboost/Adaboost) and the used features (MBLBP/Haar). In our work, best thresholds are 500 weak classifiers for the MBLBP/Gentleboost schema and 200 for the Haar/Adaboost schema.

4.3 Text localization

Once the Arabic text/non-text classifiers are learned, the next step consists in defining their application procedure into the video frames in order to detect text regions and precisely localizing text lines.

In our work, for the Boosting approaches, we use the sliding window technique. Although this method is very time consuming, the Integral Image technique, that we previously mentioned, provides a large optimization of the scanning procedure. It has been widely used in many Boosting-based detection problems in order to efficiently compute the Haar-like features and many other area-based hand-crafted features. In our work, we use it to compute both MBLBP and Haar-like features on the whole video frame before

applying the Boosting-based classifiers. Thus we do not have to compute the features at each sliding window.

Regarding the ConvNet-based detector, the sliding window technique remains a very problematic procedure. Both features and classification are learned using fixed-size images. If the problem is seen from this perspective that is strongly related to the fully connected layers, the network must be applied at each position of the sliding window in order to get the classification outputs. However, from other perspective, a ConvNet can be seen as a set of convolution layers with 1×1 convolution kernels and a full connection table. Thus, they can be trained on images with a fixed size and producing a single output vector and then, applied on larger images producing, instead, a spatial map of output vectors. The procedure goes back to the 'Space Displacement Neural Net' [MBLD91] and it is used in many detection problems like the Convolution Face Finder proposed by Garcia and Delakis [GD04]. Therefore, in our work, the video frame is convolved at once by the network. It is a pipeline of convolution and non-linear transformations applied on the entire image, reducing four times its size (mainly due to the sub-sampling). This can be seen as applying the network within a sliding window having the size of the network retina and with a sliding step of 4 pixels in horizontal and vertical directions.

In order to detect texts with different scales, we use a multi-scale text search architecture. A pyramid of different scales is constructed by up- and sub-sampling the input video frame. Our trained classifiers are then applied on the different resulting images as explained above. Then, we collect responses at each scale and group them according to their proximity in space and scale using a kmeans-like algorithm which leads to text line construction. In order to delimit left and right text borders, we proceed to a vertical histogram analysis of text line candidates. Therefore, we obtain a set of clusters, each described by the average scale and density of the group. By applying a threshold on that density, we can eliminate a considerable number of false alarms. In addition, our schema is sufficiently robust to multi-line detection giving that classifiers are trained to reject grouped lines.

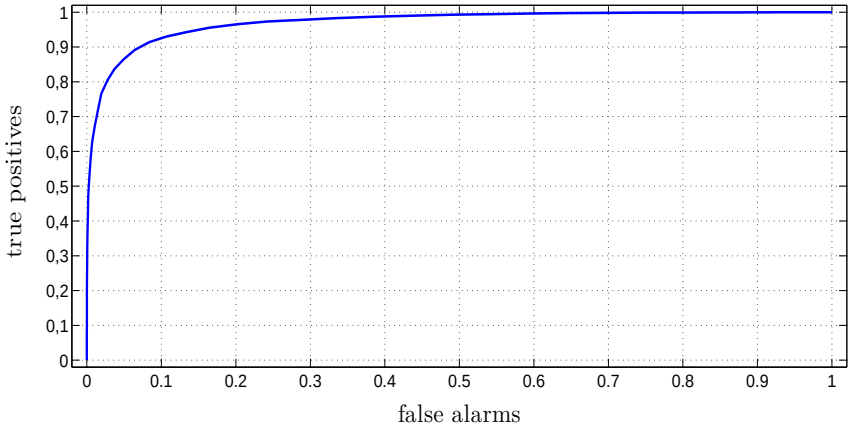
4.4 Experiments

As we previously mentioned, our detectors are trained using the *TrainDet* dataset. To evaluate the final detection results, we use the *ES1* and *ES2* sets already presented in Section 3.3 of the previous chapter and with respect to the metrics presented in Section 3.5.1 of the same chapter. Hereafter, *MLBPgentle* refers to the detection model based on MBLBP features and Gentleboost algorithm. *HAARada* refers to the model based on Haar-like features and Adaboost.

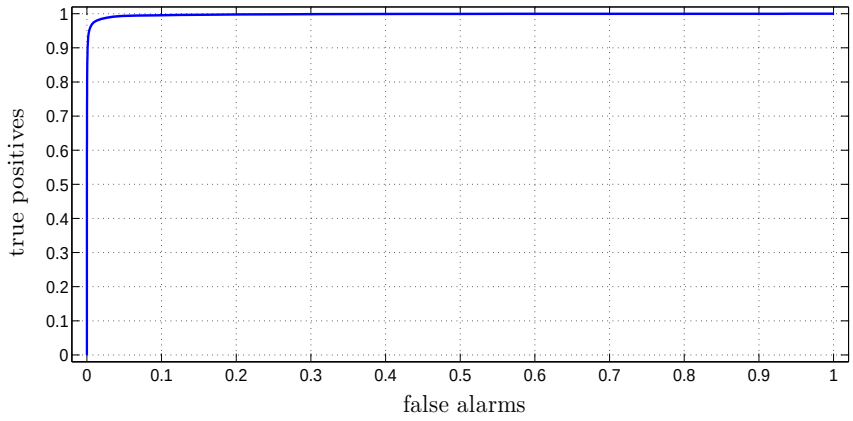
4.4.1 Evaluation of the classification models

First of all, once trained, we evaluate our classifiers performance using the *TestDet* dataset (c.f Section 3.3) which is composed of 8,000 text patches and 20,000 non-text patches. Figure 4.4 shows the resulting ROC curves of the *MLBPgentle* and *HAARada* classifiers. These two curves reflect the excellent classification capacity of the two methods. They can correctly classify more than 94% of texts with only 10% of false alarms.

The curves show also that HAARada outperforms MLBPgentle. This may be explained by the nature of MBLBP features that captures essentially large scale structures in an image which is not efficient in the case of texts.



(a) MBLBPgentle.



(b) HAARada.

FIGURE 4.4: Evaluation of Boosting-based classifiers performance on the *TestDet* dataset.

Similarly, we evaluate the proposed ConvNet classifier on *TestDet* dataset. We report a classification rate of 99.35% with the proposed architecture which clearly outperforms the classification rate of the Boosting approaches. This reflects the strong discrimination ability of the ConvNet as a classification model. Several ConvNet architectures are trained and evaluated in terms of classification accuracy over the test set. We use the same connection architecture and we change the number of feature maps on the first convolution layer n_{C1} which induces changes in n_{C2}, n_{S1}, n_{S2} and n_{N1}. Results are reported in Table 4.1. First, looking at the high classification accuracy of the three models, we remark that no over-fitting or under-fitting has been reached during training using these architectures. However, best results are obtained by the ConvNet1 model which are also very close the ConvNet2 results. However, the ConvNet2 model is still

much more lighter than the ConvNet1 model in terms of number of parameters which reduces response time. For the ConvNet3 model, with much less feature maps and neurons, the classification accuracy decreases by almost 1 point compared to the two other architectures. Therefore, a best tradeoff between accuracy and response time is achieved by the ConvNet2 model which is retained for the rest of our experiments.

TABLE 4.1: Classification accuracy of different ConvNet architectures. Architectures are presented in terms of n_{C1}, n_{C2}, and n_{N1}.

Architecture	$n_{C1} - n_{C2} - n_{N1}$	**Classification accuracy (%)**
ConvNet1	6-27-27	99.61
ConvNet2	5-20-20	99.35
ConvNet3	4-18-18	98.56

4.4.2 Evaluation of the final detectors

The proposed detectors are evaluated on both test sets *ES1* and *ES2* in terms of recall, precision, detection rates and number of false alarms.

Evaluation on *ES1*
We report the obtained evaluation results on *ES1* in Table 4.2. These results reflect the good detection capacity of the proposed methods. We can notice the excellent precision rate of the ConvNet-based method. This is due to the good rejection ability of false alarms (cf. the 5^{th} column of the table). Although HAARada outperforms the other methods in terms of recall, the ConvNet-based detector still realizes the best tradeoff between recall and precision in the one hand and detection rate and the number of false alarms in the other hand. We notice, however, the low amount of precision for the MBLBPgentle method. This can be explained by the nature of the chosen features that capture structures with diverse scales including relatively large scale ones in the image which increases the number of false alarms.

	Recall	**Precision**	**F-mesure**	**DR**	**#FA**
ConvNet	0.75	0.8	0.77	89%	**45**
HAARada	0.77	0.72	0.74	92%	170
MBLBPgentle	0.70	0.32	0.44	65%	220

TABLE 4.2: Experimental results on ES1.

Response time
In order to evaluate the speed of the proposed detectors, we evaluate the average response times over 576×1024 images[2] for each model. Results are illustrated in Table 4.3. These results show clearly that the ConvNet-based detector outperforms Boosting-based methods in terms of response time. This is mainly due to the stimulation mechanism

[2]Experiments have been conducted on a machine running Intel(R) Core(TM) I5, 2.67 GHz, 4Gb of RAM.

of large images using the ConvNet classifier which leads directly to spatial map of output vectors. For Boosting-based approaches, although a large part of computational complexity related to features is resolved using the Integral Image, several classification operations remain redundant within the sliding window technique. Results show also that the MBLBPgentle is very slow. This can be explained by the large amount of MBLBP features (compared to Haar patterns for example) which induces large amount of weak classifiers.

	Resp. time (sec)
ConvNet	7.25
HAARada	14.75
MBLBPgentle	46

TABLE 4.3: Response time of the proposed Arabic text detectors on 576×1024 images.

Evaluation on *ES2*

In order to further evaluate the generalization of the proposed methods, we conduct the same experiments on ES2. Obtained results are reported in Table 4.4. The table shows a good generalization of the three methods. There are almost no significant difference between these results and the ones obtained on ES1 (cf. Table 4.2).

	Recall	**Precision**	**F-mesure**	**DR**	**#FA**
ConvNet	0.77	0.75	0.76	97%	66
HAARada	0.75	0.66	0.70	94%	134
MBLBPgentle	0.72	0.25	0.37	85%	563

TABLE 4.4: Experimental results on ES2.

Some detection results of the ConvNet method are shown in Figure 4.5. We can see that some scene texts are also detected. This shows the generalization ability of our methods. However, it is worth to point out that these detections are considered as false alarms by our evaluation procedure. We have annotated only embedded text in our test sets.

4.4.3 Application to video indexing

Embedded text appearance in a video often indicates the beginning of an interesting sequence (e.g. the beginning of a new subject in news or the appearance of a person on screen). Text detection can be then directly used to detect such key-moments. In this work, we apply the ConvNet-based detector (our best method), in order to detect Arabic embedded text in a football match (new scores, penalty, etc.). The ground-truth is a set of segments $G = \{G_i, i = 1 \dots n\}$ indicating embedded texts. Detection results consist in a set of segments $D = \{D_i, i = 1 \dots m\}$ each containing frames with detected

FIGURE 4.5: Examples of detection results using HAARada.

texts. Recall and precision are computed as follow:

$$R = \sum_{i=1}^{n}\sum_{j=1}^{m}\omega(G_i, D_j)/n \qquad P = \sum_{i=1}^{n}\sum_{j=1}^{m}\omega(G_i, D_j)/m$$

where $\omega(G_i, D_j)$ equals 1 if the D_j and G_i overlapping is over 29 frames[3], and 0 otherwise. Obtained results show that 100% of embedded texts in the ground-truth are detected, that is our method is able to identify 100% of the highlights (recall=1). As for precision, it is equal to 79%. It is worth pointing out that 50% of reported false alarms contains sequences with scene texts that have been detected.

[3]The number of frames corresponds to the minimal duration of text appearance in the video

4.5 Conclusion

We have presented in this chapter, our proposed solutions for Arabic text detection in videos. The built detectors are able to extract text lines without any pre- processing or tedious heuristic constraints. The solutions are based on machine-learning techniques. The first detector is based on a Convolution Neural Network that learns features extraction and classification using a single neural structure. The two other detectors rely on Boosting techniques using hand-crafted feature, namely MBLBP and Haar-like features. All the challenges related to the text specifications, acquisition conditions and the background complexity are faced thanks to two factors: (1) a sophisticated training procedure that learns discrimination rules between Arabic text and non-text patterns directly from data and (2) a training dataset issued from Arabic videos that provides an exhaustive presentation of the different Arabic text and non-text modalities and their relative challenges. A special focus has been given to the rejection ability of false alarms in order to provide precise localization of text lines. This is done by training the detector to reject critical or hard non-text patterns like non-Arabic text regions and badly cropped texts.

Experimental results highlight the good detection abilities of our methods specially for the ConvNet-based method with very few false alarms. As a connectionist approach that drives both feature learning and classification in a single training process, the ConvNet is more able to explore and learn hierarchical or deep patterns from text and non-text images that fits its discriminative goal. For the Boosting-based training, using the cascaded structure allows to build stronger classifier by adding some hierarchical aspect to the training procedure. However, the reliability and efficiency of these methods sill strongly rely on the used features. This has been depicted throw the obtained detection results over Haar-like and MBLBP features. Once Arabic text lines are detected and extracted, the next step in our recognition schema consists in automatically transcribing it. This recognition task will be addresses in the next chapter.

Chapter 5

Arabic text recognition in videos

Contents

5.1 Introduction

As presented in Chapter 3, the Arabic script has many specifications that make its recognition a challenging task. One of the most difficult steps is the segmentation of the text into characters due to its cursiveness and morphological richness. In addition, handling video contents creates other challenges related mainly to the background and acquisition condition complexities. It can cause early problems if we opt for a binarization of the text zones or later during text pattern analysis.

In this work, to alleviate the cumbersome of these challenges, our recognition methodology is segmentation-free and do not rely on any pre-processing of the text images. The idea is to transform the transcription task into a temporal classification one. Using a multi-scale scanning scheme, the input text image is seen as a sequence of features without any prior information about character boundaries. The sequence labeling is

performed using BLSTM Recurrent Neural Network trained with a CTC-based objective function without any prior segmentation information. One of the main advantages of the BLSTM-CTC component is its ability to learn long-range dependencies in sequential data. Unlike HMMs, this network performs the transcription without any prior parametrization or cumbersome infrastructure referring to text elements. Although the LSTM network proved significant performance in many temporal classification tasks like speech and handwriting text recognition, it strongly depends on what it receives as features. Face to the challenges previously mentioned, finding relevant representation of text structures in video contents remains another crucial problem to focus on. Using hand-designed features and their statistical models (histograms, mean, variance...) may be a solution. However, these features are still limited by what human people think is interpretable or what they can add as complexity. Instead, we propose to learn text features directly from images. We mainly use features based on Deep Auto-Encoders (deep AE) and Convolutional Neural Networks (ConvNet). We propose also to explore the contribution of unsupervised feature learning.

This chapter is dedicated to the proposed Arabic text recognition paradigm. A global representation of the whole schema is given in Section 5.2. Section 5.3 review the different connectionist models used to learn to generate relevant text features, namely the deep AEs and ConvNet models trained in a supervised manner and the unsupervised feature generator. The sequence labeling step is then described in Section 5.4 where we review the different components used for temporal classification of text features. In this section, we give an overview of the recurrent neural networks and its variants to handle long-range dependencies in data, namely LSTMs and BLSTMs in addition to the CTC component. Section 5.5 provides all the experimental setup for building and evaluating the different models. We present and evaluate, in this section, results of the training process of the used models for feature extraction and their tuning. Then, we evaluate and discuss the recognition results on ALIF test sets. We conduct in parallel a comparative study of our methods to other recognition schema using Histograms of Oriented Gradients (HOG) and hand-crafted features and also to a commercial OCR system. Different training parameters and challenges are discussed in this section.

5.2 General overview of the proposed approach

The proposed text recognizer [YBG15b] is illustrated in Figure 5.1. The method does not rely on any pre-processing or prior segmentation. It is composed of two main steps: (1) text feature extraction from the input text image and (2) sequence or temporal labeling.

In a first step, the input text image is transformed into a sequence of relevant features. Giving the assumption that a text is a sequence of characters, we aim to find a better representation of this sequential behavior at the image level. The goal is to build, at a first step, a feature extraction model that preserves the crucial information at different levels (character, sub-characters, words, sub words, etc). To do so, we use a combination of a multi-scale scanning scheme and deep neural models to present the whole text image as a vector of learned features.

The scanning procedure is based on the sliding window technique. However, in this work we do not use a single window to sweep the input text image. As previously

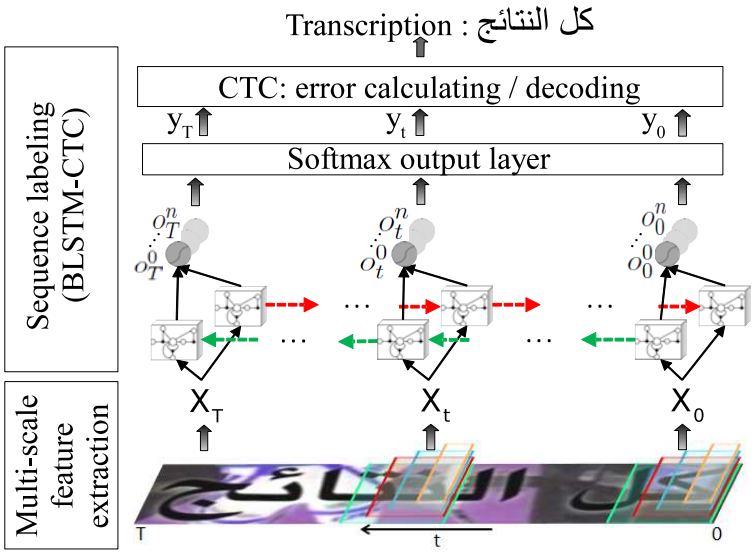


FIGURE 5.1: General overview of the proposed approach.

mentioned, extracted texts vary often in font and scale. Even in the same text image, Arabic characters have different shapes with huge variation in height and width. As a solution, we use 4 sliding windows to cover the different possible positions of characters. As shown in Figure 5.2, for an input text image of height H, we apply the scanning procedure by a step of $h/7$. Considering a set of randomly selected text images, this step value corresponds to the mean of thinnest characters widths. A set of 4 windows are applied at each time-step (scanning position) having the same height h and different widths experimentally fixed, namely $\frac{h}{4}$, $\frac{h}{2}$, $\frac{3h}{4}$ and h. Thus, different scales are taken into account.

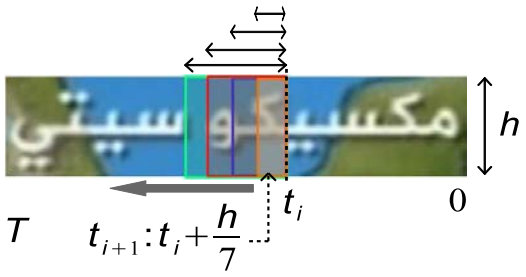


FIGURE 5.2: Multi-scale scanning procedure.

The scanning procedure gives as output a group of 4 windows at each time-step. Each of them is a local 2D view of the input text image at a given position and scale. Our goal in this stage is to find a crucial representation of the extracted sub-images that maintains relevant text structures for the recognition step. Our approach is based on learned features. The core idea is to deeply learn a function to capture data regularities while being robust to font, scale, background and noise challenges in the video. We explore four main deep learning models for a high-level hierarchy text feature extraction: Deep Belief Networks-based auto-encoders [Hin07], Multi-Layered Neural Network-based deep auto-encoder [DC93], Convolutional Neural Network [LB95] and Restricted Boltzman

Machines (RBMs) for unsupervised feature learning. The first two models learn to reconstruct each input however the ConvNet is learned with a character classification goal. These models are described in the following section.

5.3 Text feature extraction

5.3.1 Deep Auto-Encoders based text feature extraction

Auto encoders are deterministic neural networks trained with a reconstruction goal. Given an input image x, the network aims to get the same image as output. In other terms, the AE tries to learn a function f_{AE} that best maps the input to a close image of it (even for unseen inputs). For an AE, as shown in Figure 5.3, this function can be considered as a pipeline of two functions:

1. the encoding function $h = Encod(x)$ which maps the input from $\mathbb{R}^D$ to $\mathbb{R}^{N_h}$, where D and N_h are the dimensions of the input image x and the code layer respectively.
2. the decoding function $\widetilde{x} = Decod(h)$ that maps back into the input space $\mathbb{R}^D$ such that $\widetilde{x}$ is the reconstructed image of x.

In practice, the parametrization of these functions can be expressed as follow:

$$h = Encod(x) = f_E(x^T W_E + b_E) \tag{5.1}$$

$$\widetilde{x} = Decod(h) = f_D(h^T W_D + b_D) \tag{5.2}$$

where W_E and W_D are the encoding and decoding weight matrices respectively, b_E and d_D are offset functions and f_E and f_D are the activation functions.

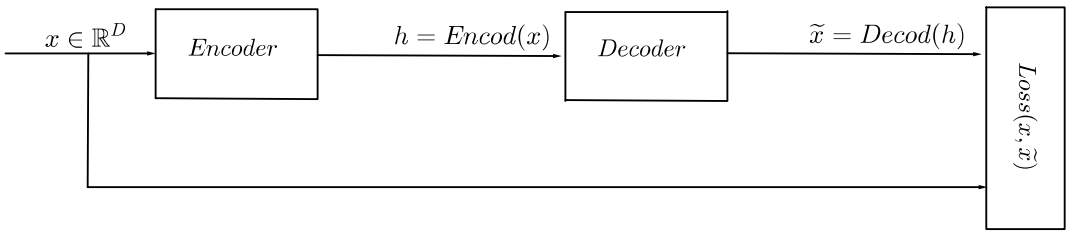


FIGURE 5.3: General Auto-Encoder structure.

Building an AE consists in finding W_E, W_D, b_E and b_D that minimize a loss function between the input and its reconstruction. This parametrization is generally achieved by a learning process using the backpropagation algorithm. AEs have been successfully used in many settings like dimensionality reduction [HS06] and image denoising [VLBM08, Cho13]. A main advantage of AE is that, during the reconstruction learning, the network can filter-out insignificant details of the input image for a better modeling of the visual object. This strongly depends on the learning procedure and the architecture of the AE itself. Using linear units and squared loss, the AE learns same subspace as PCA. This remains true for an AE with one single non-linear hidden layer,

linear outputs and squared loss. Adding non-linear layers before and after the code layer enables the representation of data that lies on non-linear manifold. Therefore, hierarchical or deep AEs allow the encoder and decoder to capture high-level abstract features of the input. Moreover, statistically, it is more efficient to learn small deep architectures (in terms of number of parameters) than shallow architectures with large number of neurons [BLP+07].

In this work, we propose to use text features learned by Deep AEs to extract relevant description of the sliding window. We focus, in particular, on two types of AEs that differ in the choice of the model's weights initialization. The first is based on a DBN that relies on an unsupervised pretraining for a better initialization of the AE parameters. This makes the model closer to the best reconstruction solution. The second AE is based on a simple MLP that starts with random weights and minimizes the reconstruction error directly and only with the backpropagation technique.

5.3.1.1 Deep-Belief Networks

A Deep Belief Network (DBN) [HS06] is a feed-forward neural network with one or more layers containing hidden units often called feature detectors. A particularity of DBN is that the learning procedure of generative weights can be layer-wise. The values of latent variables of a pair of layers can be learned at a time. This is done under the assumption that the internal states of one layer are the input data of the other one.

In this work, we use an instance of deep learning strategy: an unsupervised pre-training of Restricted Boltzman Machines (RBM) followed by a fine-tuning procedure in a supervised manner.

RBM

RBM is a probabilistic generative model. It can be seen as a Markov random field with a bipartite graph structure of stochastic visible and hidden units (cf. Figure 5.4). This structure encodes a probability distribution $p(x)$ using an energy function such as high energy is assigned to less probable states and low energy to states with high probability.

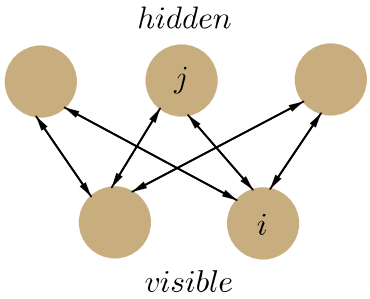


FIGURE 5.4: Diagram of Restricted Boltzmann Machine with 2 visible units and 3 hidden units.

Following the *Gibbs* distribution, this probability is expressed as follow:

$$p(v,h) = \frac{1}{Z} exp(-E(v,h)) \tag{5.3}$$

where v and h are respectively the visible and latent variables, E is the energy function and Z is the finite partition function used for normalization ($Z = \sum_{v,h} exp(-E(v,h))$).

Given two separate layers with, respectively, n_v visible binary units and n_h hidden binary units interacting through a weight matrix $W \in \mathbb{R}^{n_v \times n_h}$, the energy function is expressed as follow:

$$E(v,h) = -\sum_{i=1}^{n_v}\sum_{j=1}^{n_h} v_i h_j w_{ij} - \sum_{i=1}^{n_v} b_i v_i - \sum_{j=1}^{n_h} c_j h_j \tag{5.4}$$

where $b \in \mathbb{R}^{n_v}$ and $c \in \mathbb{R}^{n_h}$ are the offsets of the visible and hidden units respectively.

Given image pixels presented as visible units, training an RBM consists in adjusting the weights and biases such that low energy is assigned to that image and high energy to other images. This training can be achieved by the Constructive Divergence algorithm proposed by Hinton [Hin02]. The idea is to compute, first, the binary states of the hidden units, in parallel at a time, given the training visible image vector. These hidden states control the activation of visible units in the next step. Then, one more time, the hidden units are updated given the new visible states, but here, they represent features of '*reconstruction*' or '*confabulation*'. The probabilities of activating a hidden unit j given a training image v and a visible unit i given a hidden vector h are expressed by Equations 5.5 and 5.6 respectively.

$$p(h_j = 1|v) = \sigma(c_j + \sum_i v_i w_{ij}) \tag{5.5}$$

$$p(v_i = 1|h) = \sigma(b_i + \sum_j h_j w_{ij}) \tag{5.6}$$

where $\sigma(x)$ is the logistic sigmoid function ($\sigma(x) = \frac{1}{1+exp(-x)}$).

The weight update is given by:

$$\Delta w_{ij} = \epsilon(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{recons}) \tag{5.7}$$

where:

- ϵ is the learning rate,
- $\langle v_i h_j \rangle_{data}$ is the fraction of times that the pixel i and hidden unit j are activated together when the RBM is stimulated by training image pixels,
- $\langle v_i h_j \rangle_{recons}$ is the fraction of times that the visible unit i and hidden unit j are activated together when the RBM is stimulated by '*confabulation*' data.

Given this process, the hidden units can be viewed as explanatory factor of the data. In our work, we stack muliple RBMs in a pipeline fashion so that the resulting network can discover more complex data structures. Thus, the '*communication*' visible/hidden

progresses from one level to another as the network 'believe' on its own feature discovers or creations.

Auto-Encoding

In this work, a set of 4 RBMs with $\left[n_h^1 - n_h^2 - n_h^3 - n_h^4\right]$ units are pre-trained using the contrastive divergence algorithm using Arabic character images. Given that we deal with real valued input images, we replace the binary visible units of the first RBM by Gaussian visible units. In the general case of natural images, using linear units with independent Gaussian noise in the first level RBM is more representative of the data than binary units. The energy function is thus expressed as follow:

$$E(v,h) = -\sum_{i=1}^{n_v}\sum_{j=1}^{n_h}\frac{v_i}{\sigma_i}h_j w_{ij} - \sum_{i=1}^{n_v}\frac{(v_i - b_i)^2}{2\sigma_i^2} - \sum_{j=1}^{n_h} c_j h_j \tag{5.8}$$

where σ_i is the standard deviation of the Gaussian noise for the visible unit i. In our case, we use noise free reconstructions by normalizing each component of the data to have zero mean and unit variance. So the variance in Equation 5.8 is set to 1. The remaining visible and hidden units are binary except the hidden units of the top RBM which represents a low-dimensional code. Setting its hidden units to stochastic real-valued states allows a better exploitation of continuous variables.

This pre-training insures good initialization of the AE parameters for a better convergence. Higher level hidden layers achieve a dimensionality reduction and explore high-level data structures. Once pre-trained, the RBMs are 'unrolled' to produce deep encoder and decoder as shown in Figure 5.5. The resulting auto-encoder is then fine-tuned using the back-propagation of error derivatives to find the optimal character reconstruction. In this stage, the stochastic unit activities are replaced by deterministic real-valued probabilities. During training, the visible units are set to the activation probabilities of the hidden units in the previous RBM, but the hidden units of every RBM except the top ones had stochastic binary values.

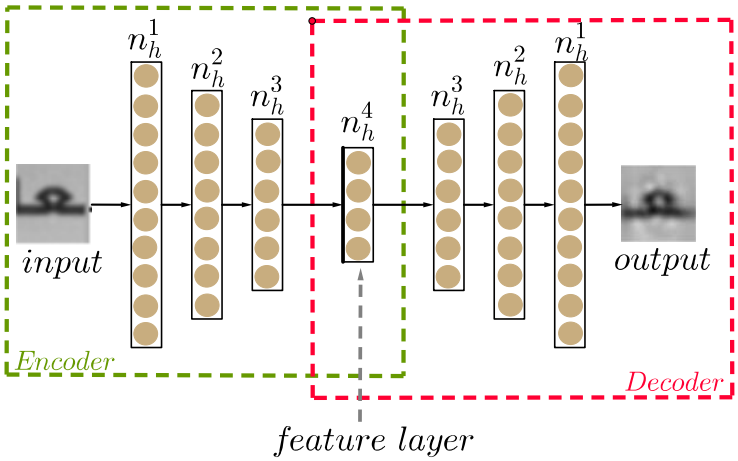


FIGURE 5.5: Deep Auto-Encoder formed by 'unrolled' RBMs.

The network is trained using the *ArabCharSet* dataset which contains basically character images with different aspect-ratios. A subset of 12% of the whole *ArabCharSet* has been dedicated for the model test. The remaining 88% has been split into 90% for training and 10% for validation. At each training epoch, the reconstruction error is evaluated on the validation set in order to control both generalization and over-fitting. The weights and bias configuration is saved if this error decreases. Once trained, the learned model is separately applied on each scanning window. Each window is then encoded by a feature vector corresponding to the outputs of the last layer of the encoder with n_h^4 values (cf. Figure 5.5).

5.3.1.2 Multi-Layer Perceptron based Auto-Encoder

An auto-encoder can be simply built using a Multi-Layer Perceptron (MLP). Given an input character image, a feed-forward network learns to produce an output that is identical to the input. In our work, as shown in Figure 5.6.b, a 3-layered neural network with fully connected hidden units is used. Employing more than one hidden layer with non-linear units in the auto-encoder enhance the ability to capture multi-modal features which specify it from Principal Component Analysis (PCA). The network is trained to minimize the Mean Square Error (MSE) between inputs and reconstructions using the back-propagation algorithm. Like DBN, the resulting encoder can be then applied to map each normalized scanning window in the new learned space defined by the output vector of the 2^{nd} layer of the encoder (feature layer).

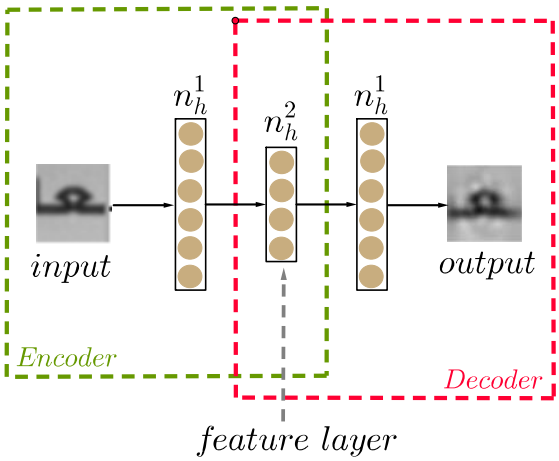


FIGURE 5.6: MLP-based Auto-Encoder.

5.3.2 ConvNet based text feature extraction

A second group of text features that we explore at this stage are based on a character classification goal. Learning features from data in general can help facing the huge complexity of video contents (background, acquisition conditions, etc). However, our final goal is recognizing texts. Therefore, a crucial point to focus on is the discrimination degree of the chosen features. Our chosen third feature generator is based on a ConvNet that learns character classification from images. A detailed description of this type of

networks has been given in Chapter 4. As previously mentioned, ConvNets as deep neural architectures have provided high level efficiency in many classification tasks. Thanks to its hierarchical structure, the network can learn complex multimodal representations of the input images. The key behind this performance is not simply the multilayered structure of the network but also the incorporation of three main hierarchical aspects namely local receptive fields, weight sharing and spatial sub-sampling. These aspects make the network very robust to shift, distortions and variations in scale and rotations.

As presented in Figure 5.7, the used Arabic character classifier in our work consists in a ConvNet of six layers excepting the input plane (*retina*) that receives an image patch of 36×36 pixels. The network aims to recognize the character presented in the image or to classify it as non-character. Layers from C_1 to S_2 correspond to a pipeline of parametrized convolution and sub-sampling operations which are applied to the input and give thus a set of *feature maps* at each layer. The first layer $C1$ contains n_{C1} maps of 32×32 pixels each, obtained by applying convolutions on the input image with 5×5 different trainable masks. This ensures a first level feature extraction. The 2^{nd} layer $S1$, with $n_{S1} = n_{C1}$ feature maps of 16×16 pixels, is a sub-sampling layer of previous maps. This reduces their sensitivity to shifts, distortions and variations in scale and rotation. These first two layers are in charge of a first level (or simple) feature extraction. The 3^{rd} layer ($C2$) applies both feature combination and a 3×3 convolution on $S1$ with n_{C2} feature maps of 14×14 pixels. The 4^{th} layer ($S2$) is similar to $S1$ with $n_{S2} = n_{C2}$ maps of 7×7 pixels. These two layers perform second level feature exploration (high-order features). These low dimensional disjoint and steady features are fed to a classical MLP of two layers N_1 and N_2 that perform feature classification. The layer N_1 contains n_{N_1} sigmoid neurons that are fully connected to the feature maps of the previous sub-sampling layer S_2. Each neuron of N_1 is connected to all units of all feature maps of S_2. It performs the dot product between its inputs and connections' weights and passes the result through a sigmoid activation function. These neurons are fully connected to the output N_2 layer which is a softmax layer with 81 neurons that performs classification.

The network is trained using the *ArabCharSet* dataset with the same partitions as the previous models (in terms of training, validation and test examples). The network parameters including connections weights and bias are optimized using the back-propagation of error gradient with momentum. The learning is performed in several epochs. At each epoch, a random class-balanced selection of training images is processed by the network. For each example x, the MSE between target and output vectors (tg and out) is calculated:

$$MSE = \frac{1}{N2} \sum_{i=1}^{N_2} (tg_i(x) - out_i(x))^2 \tag{5.9}$$

where tg_i is set to 1 if i corresponds to the character class of x, otherwise, it is set to 0. An evaluation of the classification error is performed on the validation set at each training epoch. The model configuration is saved if the error decreases.

As described in Chapter 3, Arabic letters can change in form according to their positions in a word (beginning, middle, end or isolated). Thus for the classification task, we do not consider only the 28 atomic letters, but, instead, we take into account character glyphs or forms. If there is a drastic difference between two or more glyphs of the same letter, we consider them as two different classes as shown in Figure 5.8. For this example, the character 'Jiim' (ج), has basically 4 glyphs according to its position in the word. The

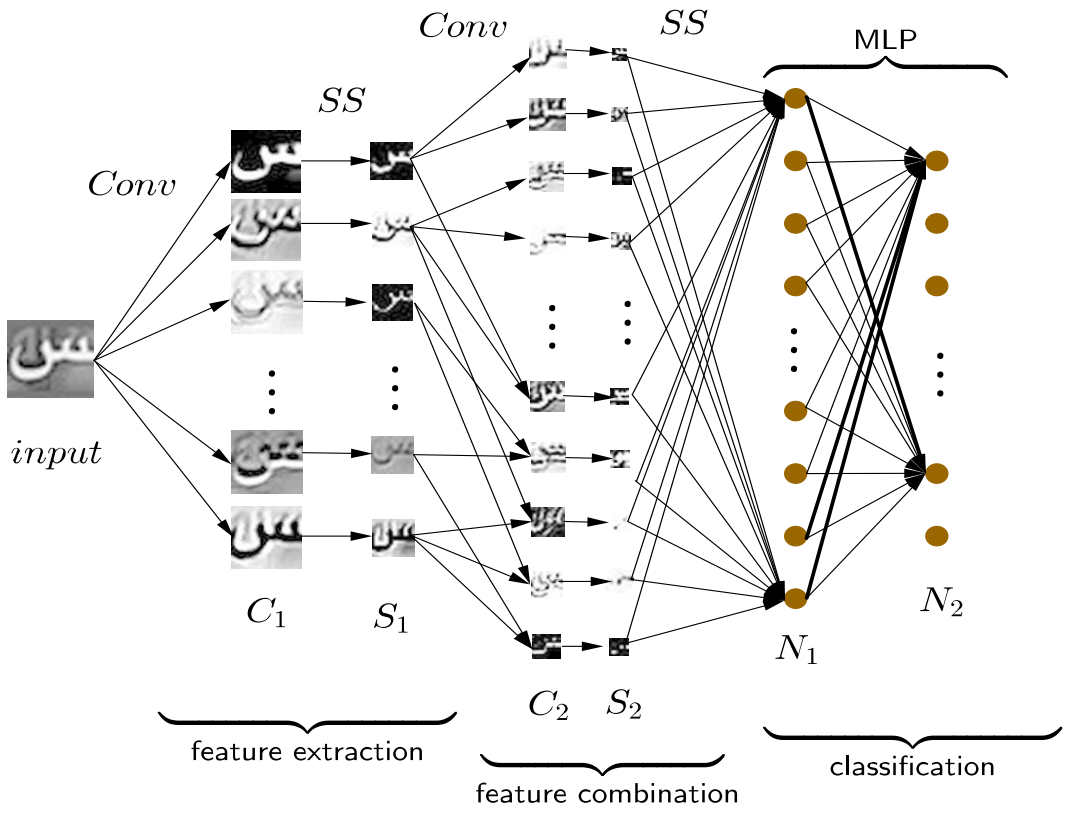


FIGURE 5.7: ConvNet-based character classifier. 'Conv' and 'SS' refer to convolution and sub-sampling operations respectively.

glyphs corresponding to the end and isolated positions are morphologically very close and similarly for the glyphs of the beginning and middle positions. Therefore, for the ConvNet, 2 classes are associated to this character. Indeed, at this stage, our goal is not to recognize characters but to capture features that represent a kind of signature of each character as a glyph. One the one hand, separating between different forms of a single character makes classification simpler. On the other hand, handling each character form as a separate class, even for very close glyphs, can be cumbersome for the network specially that the position information of a character depends, in a large part, of the context and can be inferred later in the sequence labeling stage. Consequently, at this level, the ConvNet considers 81 classes including the obtained letter glyphs, 8 punctuation marks ('.', ',', ':', '!', '?', '-', '/', '%') and the 10 Arabic digits.

Once the network is trained, it is used to stimulate text regions. We apply only the first 5 layers in order to get the outputs of the layer N_1 with a sigmoid activation function. These outputs represent the feature vector of the input image. Indeed, this layer is fully connected to the last sub-sampling layer and combines, therefore, multiple stable disjoint features into single neurons.

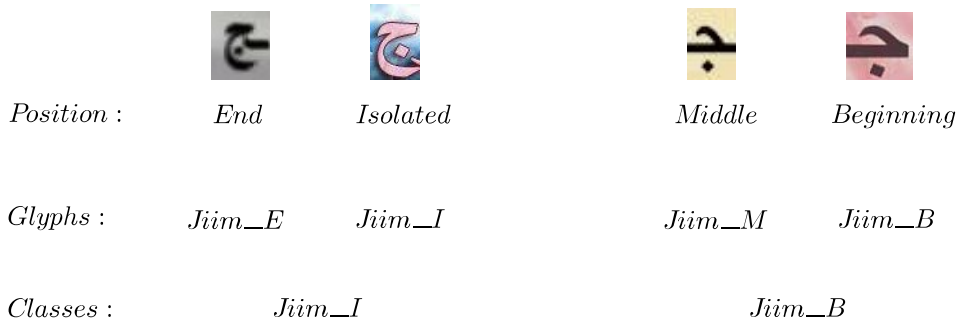


FIGURE 5.8: Glyph-based classification. Example for the letter 'Jiim' where two classes are considered corresponding to the most morphologically different groups of glyphs.

5.3.3 Unsupervised feature learning

Although the previously proposed models are different in terms of architectures and training goals, they still share one important point, namely the supervised criterion of the training process. For the deep AEs, the network defines its targets as the input images while for the ConvNet, targets consists in the character classes. In order to avoid the use of pre-labeled data, we propose a third type of features based of stacked RBMs. For the proposed DBN-based AE, we used these RBMs as a pre-training procedure to initialize weights. Here, we propose to directly use these pre-trained RBMs as a feature generator and study to which extent these features are efficient for Arabic text recognition. We use the same RBM stack introduced in Section 5.3.1.1 with the same training parameters but without any fine tuning.

5.4 Temporal classification for sequence labeling

At this stage, each text image is seen as a sequence of learned features without any priors about characters' boundaries. The only available information is that the length of this sequence is greater or equal to the target text transcription. Therefore, recognizing a text requires, in the one hand, a certain knowledge about the structure of these features regarding text elements (characters or sub-characters or even words, etc). In another terms, it requires feature-based models for these entities for further classification. In the other hand, recognition in this context relies on the definition of a mechanism that can decide when the classification should be made while processing the feature sequence. The problem is no more seen as simple pattern or segment classification but as temporal classification issue. Solving this problem can be achieved using Hidden Markov Models (HMM) [Rab89, Ben99] or Recurrent Neural Networks. In this work, our sequence labeling method relies on a special class of RNNs: the Bidirectional Long-Short Memory networks (BLSTM) coupled with a Connectionist Temporal Classification component (CTC). In the following sub-section, we will give an overview of the RNNs and the used BLSTM-CTC network while highlighting the major differences between this connectionist approaches and HMMs.

5.4.1 Recurrent Neural Networks: An overview

Since their appearance, Artificial Neural Networks (ANN) have been widely used for pattern classification tasks where the network output depends only on the current input. Many ANN models have been developed under this assumption like MLP [Bis95, Wer88b, RHW85], Kohonen maps [Koh90], Hopfield Nets [Hop82], ConvNets [LB95], etc. However, all these well-known networks are only part of the whole varieties of ANNs and clustered, more precisely, as Feedforward Neural Networks. For this type of ANNs, only acyclic connections are permitted between layers (connections feeding forward from one layer to the next). Another class of ANNs, that began to draw the attention of the machine learning community, includes neural networks with cyclic connections and are referred as *feedback* or *recurrent* neural networks (RNNs). Unlike, Feedforward Neural Networks, RNNs are able to perform temporal classification and infer outputs that depend on the current input and a certain range of its context or history. In addition to the forward connections, a RNN introduces a recurrent link which is actually a time delay connection. Figure 5.9 illustrates a simple architecture of RNN with one hidden recurrent layer fully connected to itself and to the inputs and output layers.

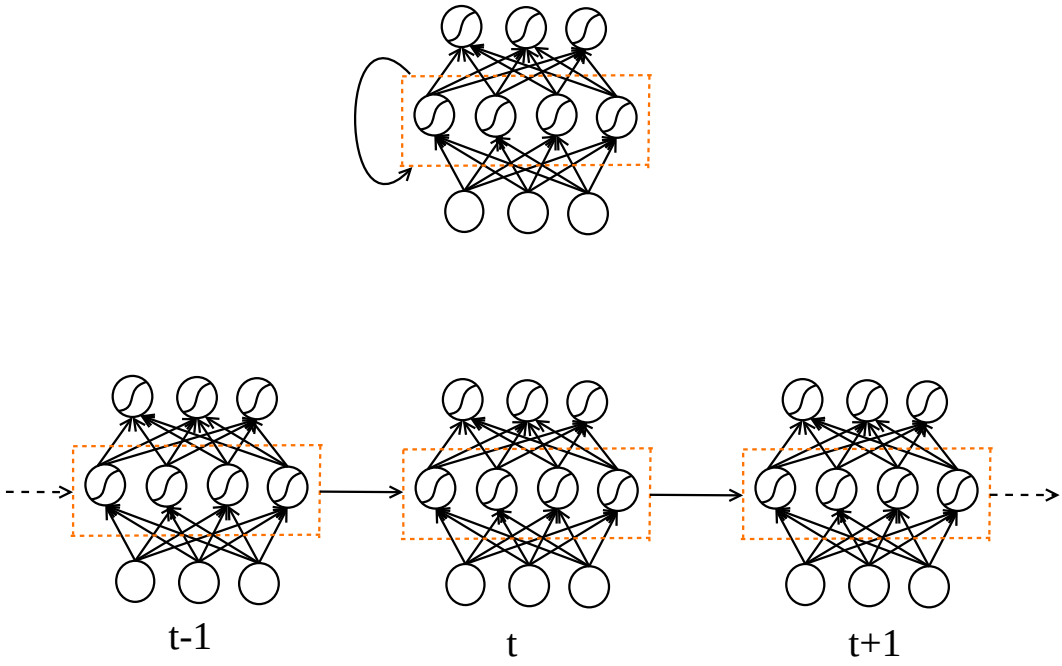


FIGURE 5.9: Architecture of a simple RNN with one hidden layer. Both compact (top) and unfolded in time (bottom) views are presented.

A recurrent layer is defined by a group of units with recurrent connections i.e. the activation a_j^t of a hidden units j at time t depends not only on the network's inputs x_i^t at t but also on the activations of hidden units at $t-1$, denoted a_k^{t-1}. Thus, for a RNN having N inputs and K hidden units with differentiable activation function σ, a_j^t is expressed as follow:

$$a_j^t = \sigma\left(\sum_{i=1}^{N} w_{ij} x_i + \sum_{k=1}^{K} w_{kj} a_k^{t-1}\right) \tag{5.10}$$

This recursion in the network connections allows to develop a kind of *memory* of previous inputs in the internal state of the network. The network outputs can be calculated in parallel at each time t taking as inputs the hidden layer activations $a_j^t,\ j \in \{1, \ldots, K\}$. It can be seen, therefore, that the outputs are implicitly influenced by both the current network's input and its *history*.

Unfolded in time, a RNN is seen as a Feedforward Neural Network. However, its training is much more complex given this time-dependent influence of the hidden units on the objective function. An efficient way to propagate errors' gradients back in time through the recurrent weights is to use the Backpropagation Through Time algorithm (BPTT) as described in [RHW86], with a practical implementation described in [Bod02]. It consists in an application of the chain rule like in standard Backpropagation algorithm. However, BPTT considers both architecture (layer level) and temporal influences of hidden units on the objective function i.e. at a time-step t, the hidden layer affects the objective function through the next layer (output layer in the simple case) at t and through the hidden units at the next time-step $t + 1$.

Given this basic RNN architecture, the network can use past contexts in order to infer actual state. In some tasks like speech and text recognition or language processing, both future and past contexts are important for sequence labeling. To tackle this issue, a special RNN architecture, called Bidirectional RNN (BRNN), has been proposed by Schuster and Paliwal [SP97]. The idea is to replace the recurrent hidden layer by two recurrent hidden layers: a forward layer that handles the input sequence in the direct time direction (from the past to the future) and a backward layer that process the sequence in the opposite direction (from the future to the past). Both layers are not connected to each other but they are connected to the output non recurrent layer. Therefore, at each point, the network output depends on both past and future contexts. Figure 5.10 illustrate the BRNN architecture.

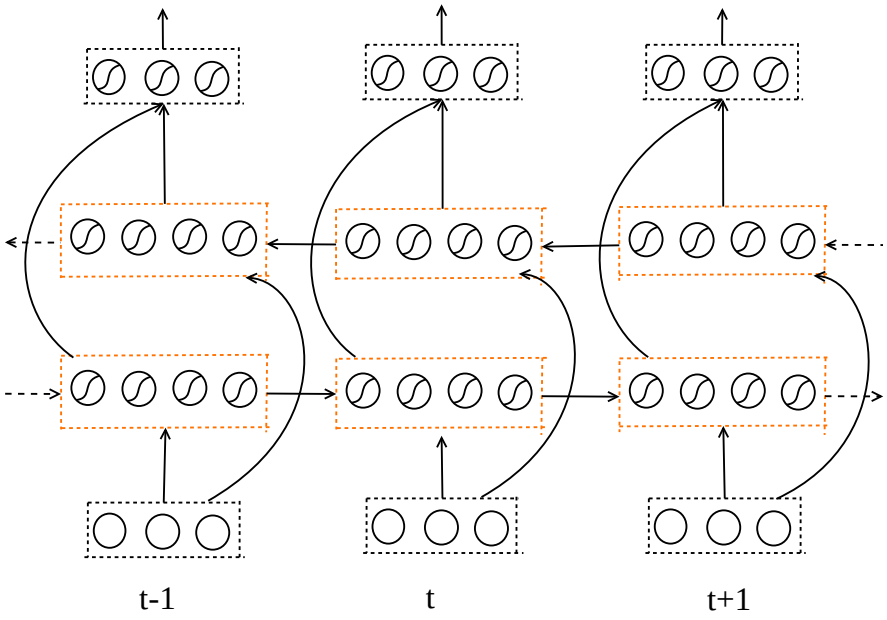


FIGURE 5.10: Architecture of a simple Bidirectional RNN.

Similarly to the RNN, training a BRNN can be achieved by the BPTT algorithm. However, for BRNN, there is a special chronological order to respect in the calculation of activations and gradients regarding output layer and each of the hidden layers. For a BRNN, all activations of the hidden layers must be calculated and stored for all processing time steps. We refer to these activations by A^t and $\widetilde{A}^t$ for forward and backward layers respectively which are vectors of hidden units activations given by Equation 5.10. Once stored, the output layer uses both of these activations at each time step to determine the network outputs Y^t. For the backpropagation step, refereed as backward pass, we denote by δ the derivative of the objective function with respect to a unit signal. For this pass, all δ terms relative to the network output layer are all first calculated and stored for all time-steps. Then, these terms are used to perform the backward pass for the forward hidden layer and backward hidden layer separately using also the chain rule like in simple RNN.

5.4.2 LSTM Recurrent Neural Networks

Although RNNs are suitable networks for modeling time series and have many advantages over HMMs, training such networks is not a trivial task given two main problems: *vanishing gradient* and *exploding gradient*. Theses issues have been revealed in details in [BSF94]. They are related to different behaviors of long term components. For *exploding gradient* problem, these terms can increase exponentially and explode during training leading to an inflation of the norm of the gradient. The issue of *vanishing gradient* is the opposite case where long term components decrease exponentially which makes hard for the RNN to learn long range correlations between slices of data. As new inputs can overwrite hidden unit activations, it is difficult for the network to maintain the first seen information in its internal states. In practice, the *vanishing gradient* occurs for tasks including delays of more than 10 time-steps between given input and the corresponding target output.

This makes simple RNNs not suitable for our problem given that for the text sequences, a set of two or three characters may take more than 10 feature vectors among the whole feature sequence. In order to face the *vanishing gradient* shortcoming, we choose to use the Long-Short Term Memory architecture (LSTM) instead of simple RNN. This special model of RNNs proves to be the most effective solution to learn long term time series correlation. It was first introduced by Hochreiter and Schmidhuber [HS97]. The core idea of LSTM is to replace non-linear hidden RNN units by recurrent connected memory blocks. Each block contains one or many memory cells. A memory cell is a special type of linear unit with a self connection of value 1. The flow of information into and out of the unit is controlled by three multiplicative units: the input, output and forget gates. The most crucial point about these units is that they are learned i.e. the control of this flow of information is learned from data.

A graphical representation of the LSTM memory block with one cell is illustrated in Figure 5.11. The different introduced gates allows controlling access to the information and protect it from perturbations during training which avoid the gradient vanishing. The input gate is the switch of reading while output is the trigger to writer. A cell in the hidden layer cannot receive the information unless the input gate is open. Similarly, it can communicate its activation to other cells only when the output gate is unlocked. This ensures to store information for arbitrary time lags but in the same time, this may produce an unbounded increase of the cell state. However, for many tasks, theses

states need to be occasionally reset (like in the case of beginning of new text example). Thus, the forget gate has been introduced [GSC00] to handle this issue. As long as this button to forget is activated and the input gate is closed, the memory cell still remembers the first input. In order to allow all gates to inspect the current cell state, 'peephole' connections have been added to the LSTM block [GSS03] (between the cell and all gates) as shown in Figure 5.11.

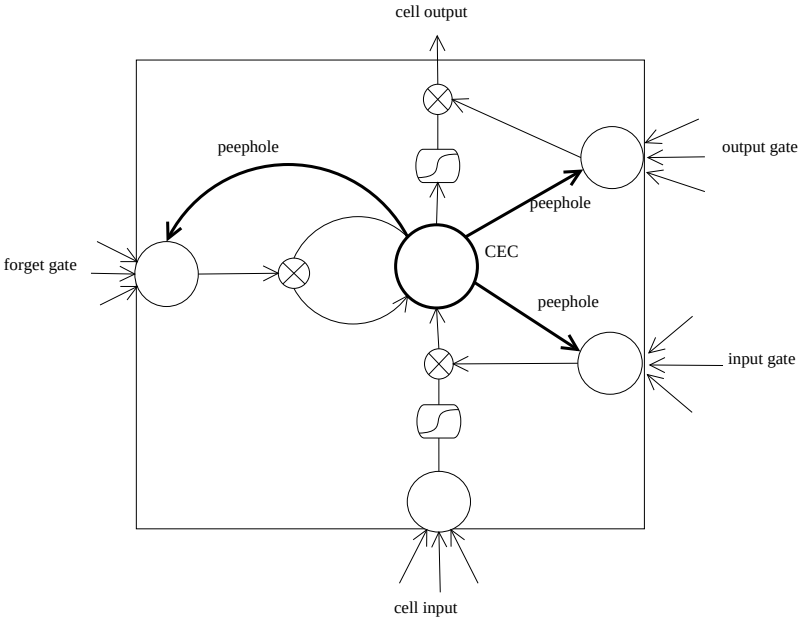


FIGURE 5.11: Graphical representation of a LSTM memory block with one cell.

LSTM networks have outperformed simple RNNs in many tasks that require long range context information like in speech recognition [GS05, GJM13, GFS05], handwriting and printed text recognition (cf. Section 2.4), text and music generation [Gra13, ES02] and language modeling [SSN12]. Like for BRNNs, bidirectional extension of LSTM networks (BLSTM) has been proposed in order to take backward information into consideration [GS05, WEG+10]. It consists of a BRNN with LSTM hidden cells instead of non-linear hidden neurons in both of the forward and backward layers. The BLSTM can be trained using the BPTT algorithm. An exhaustive presentation of the LSTM equations and the training process can be found in [Gra12b].

5.4.3 Connectionist Temporal Classification

Although LSTM-based RNNs are powerful networks for modeling sequential data, their effectiveness in temporal classification tasks still relies on prior knowledge about the sequential structure of data in order to perform training. For text transcription, given the training sequences of textual features, the output layer of the BLSTM presents, at each time step, a probability distribution over the set of character classes. This distribution presents the BLSTM classification result of the current observation. The goal of training is to get a distribution corresponding to the target character classification at each time step. Therefore, the network needs pre-segmented text images in order to determine the vector of errors at each time step. This problem has been deeply reveled

in [SH94, Gra12b]. Many works have proposed a solution to this problem by combining HMMs and RNNs to perform temporal classification [SR96, Rob94, GFS05]. The neural network role is to introduce contextual information and perform local classification task. The HMM is used for an automatic segmentation of the input sequence during training. It also performs transformation of RNNs classifications into labels or characters, a kind of alignment between classification outputs and the targeted sequence. This use of HMMs envolves the necessity to make assumptions about data and limits RNNs to a local classification task instead of a complete temporal classification process. Thus, instead of exploiting the advantages of RNNs over HMMs in such task, we inherent the system all the drawbacks of HMMs and their cumbersome infrastructure and parametrization.

The Connectionist Temporal Classification (CTC) component, has been introduced by Graves et al. [GFGS06, GLF+09, FGS07, GLB+08] in order to overcome these problems. The role of this component is to determine the temporal error vectors to backpropagate through the RNN or the LSTM using only unsegmented data. All what is known is that the target sequence is shorter than the input sequence. It does not require the integration of HMMs to represent sequence modalities nor a post-processing step to derive labels from the network outputs. It performs directly an alignment between the LSTM output sequences and the corresponding target transcription without any external parametrization. The goal is not to obtain a correct network prediction when exactly the corresponding input occurs. The goal, instead, is to obtain the overall correct transcription (all desired labels in the same order). During training, this alignment allows the network to learn the relative location of labels compared to the whole transcription. Desired LSTM outputs should represent, after training, probability peaks corresponding to target label in the same order.

To perform alignment during training, the CTC introduce an additional class 'BLANK' class that is inserted at the beginning and the end of each target transcription and between each couple of its characters. This '*BLANK*' defines the network target for inter-characters cases and also avoids the elimination of words with character repetitions during decoding.

Given an input sequence x and its label transcription l, the goal of CTC is to maximize the conditional probability $p(l \mid x)$. It should thus determine the error vectors to backpropagate at each time step. For the CTC, the objective function defined over a training set S, in the logarithmic scale, is expressed as follow:

$$\sum_{(x,l)\in S} ln(p(l \mid x)) \tag{5.11}$$

Directly computing these probabilities involves browsing all paths defined by LSTM outputs that correspond to l for each training example in S. However, it has been shown that the error that minimizes this objective function can be determined by the 'Forward-Backward' algorithm used for HMM-based decoding [Rab89, GFGS06]. A detailed description of the process can be find in details in [GFGS06] for the CTC use case. Globally, the error is based on the definition of two temporal variables : the forward variable α_t and the backward variable β_t, defined for each position s in the label transcription l and at each time-step t as follow:

$$\alpha_t(s) = p_t(l_{1:s} \mid x) \tag{5.12}$$

$$\beta_t(s) = p_t(l_{s+1:|l|} \mid x), \tag{5.13}$$

where $l_{1:s}$ corresponds to the first s labels in l and $p_t(l_{1:s} \mid x)$ is the probability of getting, at time-step t this sub-sequence of length s as output. For each character or label k and each time-step t, the error is calculated as follow:

$$e_t(k) = \frac{\sum_{s \in lab(s,k)} \alpha_t(s) * \beta_t(s)}{\sum_{s=1}^{|l|} \alpha_t(s) * \beta_t(s)}, \tag{5.14}$$

where $lab(s, k)$ refers to the set of indexes s where l_s corresponds to the character k.

These error vectors are then backpropagated through the network and all parameters are rectified in a way to redress the output probability distributions as shown in Figure 5.12 i.e. obtaining probability peaks corresponding to strong prediction of the correct label.

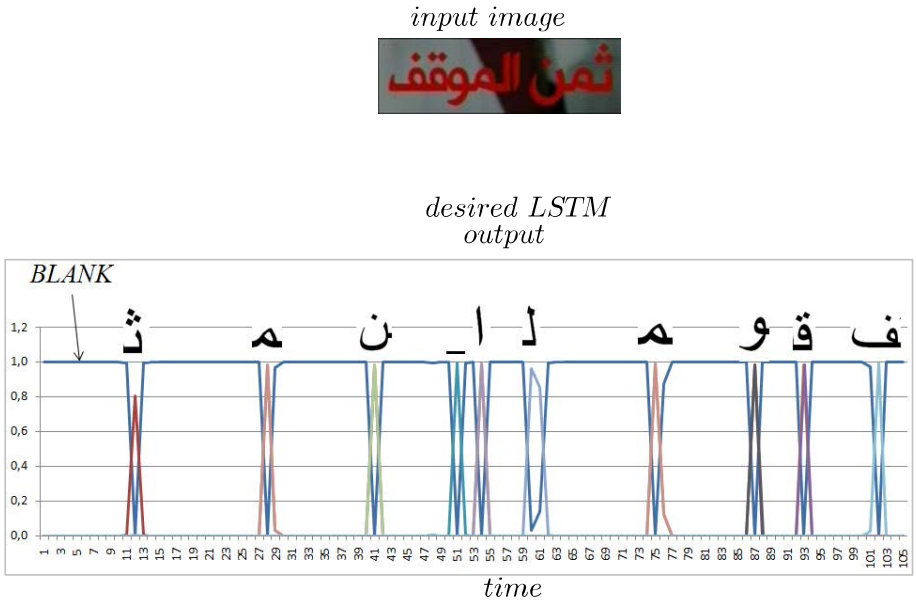


FIGURE 5.12: LSTM/RNN target output units activation through time. Curves represent the activation level through time of a set of label classes (label probability). '_' corresponds to the space character. Most of times the network emissions correspond to the 'BLANK' class.

In addition to the ability of CTC to train RNNs without segmented targets, it allows the network to focus entirely on searching for the correct labels. Such models are usually referred to as discriminative models and have many advantages over generative models like HMMs. The difference between discriminative and generative models have been deeply tackled by Bishop [Bis01]. The RNN-CTC as discriminative model, is able to directly calculate the posterior class probabilities $p(class \mid x)$. However, generative models determine first the class conditional densities $p(x \mid class)$ and uses the Bayes' theorem to infer this posterior class probabilities. This implies that for generative models, each class must be modeled independently while for discriminative models, the overall training

mechanism focuses directly on determining this separation between classes. In addition, the introduction of CTC allows to take complete advantage from the RNN abilities. We note, mainly, the capacity of this latter to model non-linear inter-dependencies between input features. In practice, HMMs generally use mixture of diagonal Gaussian to model the distribution of input features, which makes it limited to decorrelated or locally inter-dependent inputs. Another advantage of RNNs is their ability to take into account the context to estimate the probability of each observation.

5.4.4 Training architecture and sequence labeling

In our work, we use a BLSTM-CTC network for temporal classification of the learned feature sequences. The network takes as input a sequence of at maximum $T = 200$ feature vectors, each of 400 values normalized between -1 and 1. As already shown in Figure 5.1 (cf. Section 5.2), the input is fully connected to the forward and backward LSTM hidden layers of 300 cells each. This size is empirically set up (cf. Section 5.5.2). Both of these layers are connected to a third non-recurrent layer where the network combines both future and past contextual information. The output layer is a softmax function that produces predicted character probabilities for each time-step and character. At this stage, each letter at a specific location in a word is considered as a class (label). Hence, we have 130 classes (including letters, digits and punctuations) and an additional class 'BLANK' for no character case. The network is trained using the Back-Propagation Through Time algorithm with a learning rate of 10^{-4} and a momentum equal to 0.9. For training, we use the ALIF_Train text images. This dataset has been augmented to **7,673** by applying some image processing operations like color inversion, blurring, etc. Nearly 9% of the resulting subset has been used for validation. During training epochs, we evaluate both objective function and the Edit distance between recognitions and target labels on the validation set in order to ensure generalization and avoid over-fitting.

Once trained, a CTC decoding scheme is applied on the BLSTM softmax outputs using the best path decoding algorithm. After removing all the '*BLANK*' responses and successive repetitions, the most likely sequence of labels is hence deduced by just taking labels with maximal emissions at each time step. This decoding process is applied only at this stage. A more sophisticated algorithm will be applied later in Chapter 6 allowing joint decoding with a language model to improve recognition results.

5.5 Experimental set-up and results

In order to better evaluate the proposed solution, we conduct different experiments on both feature learning models and the final text recognizers.

5.5.1 Performances of feature learning-based models

The different AEs and ConvNet models for text feature learning are built using the *ArabCharSet*. A subset of 12% of the whole set has been used to test the models. The remaining 88% has been split into 90% for training and 10% for validation. For the AE, we measure the Mean Square Error (MSE) between images and their reconstructions in order to control the efficiency of the models. For the ConvNet, the Classification Error

Rate (CER) is used as evaluation metric. At each epoch of the training, the updated model is evaluated on the validation set in order to control both generalization and over-fitting cases.

5.5.1.1 Auto-Encoding

For the DBN-based AE, four RBMs with $[2000 - 1000 - 500 - 100]$ units are pre-trained for 100 epochs in a mini-batch mode. We clamp 28×28 character patches into the first 2000 visible units and a layer-wise update of the RBM weights takes place using the constructive divergence algorithm. Learning rates are set to 10^{-2} for RBMs with binary hidden units and are two orders of magnitude smaller when using stochastic real-valued units, namely for the last RBM. After pre-training, the RBMs are unrolled to form the AE that learns character images reconstruction. At each epoch of the training, we measure the reconstruction MSE on the validation set in order to track the convergence of the leaning process. The variations of this metric are presented in Figure 5.13.

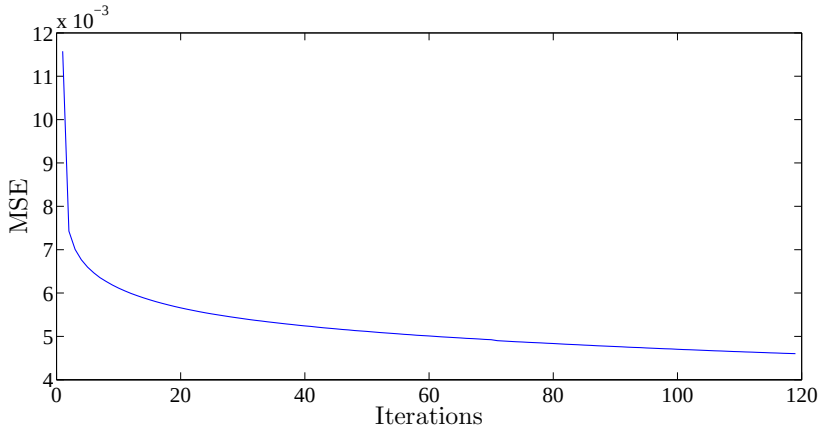
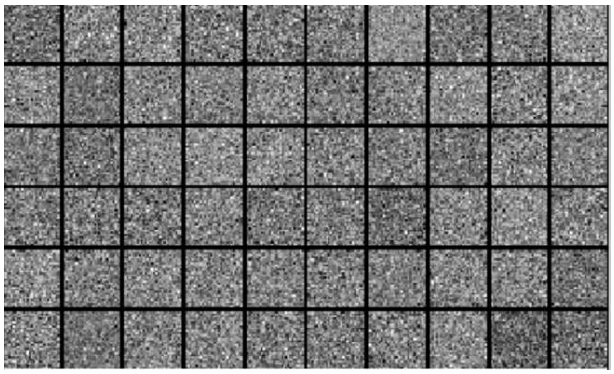


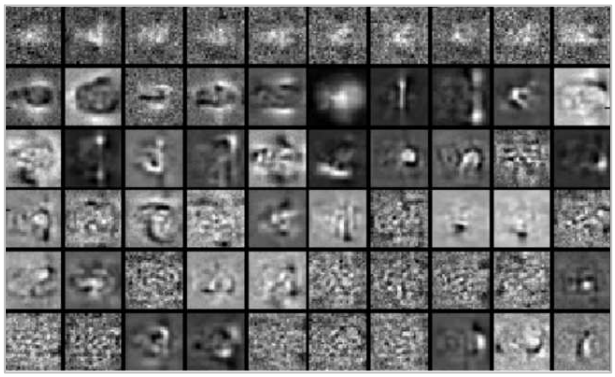
FIGURE 5.13: MSE variation during DBN training evaluated on the validation set.

In order to have an idea about what the AE extracts as features, we are interested in the representation of the weights between different layers. Particularly, weights that connect the input to one unit in the first hidden layer can be presented as a patch and are interpreted as filters. We refer to these weighs as feature detectors. Similarly, weights that connect the output to the last hidden layer are referred to feature generators and can be also presented as patches. We present in Figure 5.14 some of these feature detectors at the beginning of the training process (cf. Figure 5.14(a)) and during training (cf. Figure 5.14(b)). These features correspond to the weights between the input layer and 60 hidden units selected from the first hidden layer and are presented as patches having the same size of the input images. The evolution of the features from one training epoch to another shows that the AE develops and learns his own filters in order to explore data modalities for a better reconstruction.

After training, the DBN has been evaluated on the test subset, reaching hence an MSE of 3.5×10^{-3}. Figure 5.15 illustrates some test examples with their reconstructions. The presented images show, globally, good reconstruction results for most of characters. However, for some letters, the reconstruction yields, sometimes, a blurring effect on



(a) Before training.



(b) During training.

FIGURE 5.14: Selection of feature detectors in a DBN. These features correspond to weights connecting the inputs to 60 hidden units randomly selected. (a) illustrates features before the training process and (b) presents the evolution of these filters during training.

minimalistic structures like points and diacritics and poor reconstructions for examples with complex background. Visually, this do not lead to very serious errors but can induce some confusions between characters with one, two or three points, or between characters having points and 'Hamza' as diacritic like 'و' and 'ؤ'. However, at this level, we do not know to which extent these visual observations much text recognition results.

Similarly, for MLP-based AE, an architecture of [120-100-120] fully connected neurons has been used. The learning rate has been fixed to 10^{-3}. Initially, we train different architectures varying the number of neurons of each layer and the number of layers. Results in terms of MSE on the character test subset are reported in Table 5.1. Best reconstructions have been obtained with the MLP1 architecture. Decreasing the number of units in the feature layer can impact results significantly as shown with the MLP3 architecture. With the MLP2 architecture, we tried to decrease the number of parameters of all layers but this induced degradation of reconstruction results. Similarly, no improvement has been made when adding more hidden layers. Therefore, we retain the first architecture as model for feature extraction.

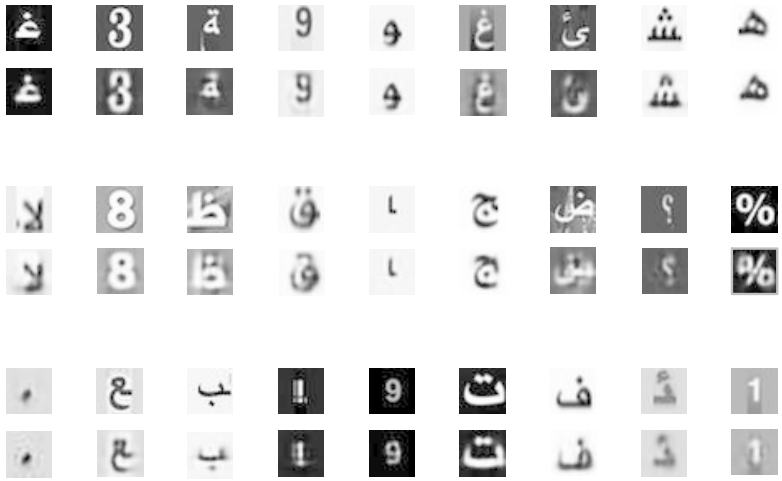


FIGURE 5.15: Examples of test character images with their reconstructions. Top rows presents original images and bottom rows presents reconstructed images.

TABLE 5.1: Reconstruction MSE for different MLP-based AEs.

	MLP Architecture	MSE $\times 10^{-3}$
MLP1	120-100-120	5.67
MLP2	80-60-80	8.52
MLP3	100-50-100	11.61
MLP4	100-80-60-80-100	6.73

5.5.1.2 Character classification

As presented in Section 5.3.2, the ConvNet is trained to classify character glyphs. We consider in total 81 character forms presented at the level of the softmax layer of the network in addition to a 'Rubish' class for non-character images. The output layer presents the probability distribution of one character patch input after simulation by the network. Different architectures ave been trained and evaluated in terms of Classification Error Rate (CER) on the test subset of the *ArabCharSet*. We consider different layer sizes for the C_1, C_2 and N_1 layers. Results are reported in Table 5.2. As shown in this table, best classification results are obtained by the Conv4 architecture. However, compared to the Conv3, the results are very close with a gain of 20 neurons in N_1 and more than 1300 trainable parameters per additional neuron. We have expanded also the MLP part in the ConvNet to 100-40-100 in order to evaluate a architecture with few neurons in the code layer. The classification results remain under the Conv3 results. Therefore, given these results, we retain the Conv3 as model for further text feature extraction. For the obtained results, characters with very close shapes present higher CER specially for characters that differ by points and diacritics like the 'Xaa'/'Haaa' (ح/خ) and 'AlifBroken'/'HamzaAboveAlifBroken' (ى/ئ) letters.

TABLE 5.2: Classification results for diffrent ConvNet architectures.

	ConvNet Architecture	CER (%)
Conv1	5-5-20-20-100	5.73
Conv2	6-6-27-27-80	4.05
Conv3	6-6-27-27-100	3.8
Conv4	6-6-27-27-120	3.64
Conv5	8-8-44-44-100	4.98
Conv6	12-27-100-40-100	7.12

5.5.2 Performances of sequence labeling: Text recognition

As previously described, the learned models of AEs and classification are applied separately on the different subsets of the ALIF dataset in order to transform text images into sequences of learned features. We train, then, the BLSTM-CTC network as indicated in Section 5.4.4. Each learned BLTM model is then evaluated on ALIF test sets in terms of recognition metrics presented in Chapter 3 (Section 3.5). Depending on the models and features, the 3 models are denoted as follow:

- ***DBN-AE-BLSTM***: DBN auto-encoder + BLSTM.
- ***MLP-AE-BLSTM***: MLP auto-encoder + BLSTM.
- ***ConvNet-BLSTM***: ConvNet Classifier + BLSTM.
- ***Unsupervised-BLSTM***: Unsupervised training with RBMs + BLSTM.

The obtained results on the ALIF_Test1 set are reported in Table 5.3. These results show high rates with the first 3 features. The proposed deep neural networks are hence able to extract relevant features that are well used by the BLSTM. They also show that the *ConvNet-BLSTM* method outperforms the two other ones by almost 4% in terms of CRR and more than 12% in terms of WRR. This can be explained by the specifications of diacritics in the Arabic text. As shown in Section 3.2, some characters are distinguished only by diacritic marks. Using the reconstruction paradigm, these marks are often reproduced with blurring effect which introduces confusions (especially between letters with 2 and 3 points). The ConvNet is a powerful classification model. It focuses only on predicting the correct character class and derives features that discriminate between characters. In particular, the conception of the ConvNet filters through layers and their learning procedure ensure some degree of shift, scale, and distortion invariance. Regarding auto-encoders, they extract features that allow reconstructing the whole input image, the character part but also the noise and the background.

Although LSTM is a discriminative model, it still strongly depends on what it receives as features. Here, it is clear that the problem of classification is partly resolved by the ConvNet that acts locally on the text image. The LSTM role is then almost restricted to temporal classification or to feature correlation learning in order to infer labeling patterns. However, integrating AE-based features reflects a part of the LSTM efficiency in handling both local and global temporal classification. The obtained results show also

TABLE 5.3: Recognition results on **ALIF_Test1**.

	CRR (%)	WRR (%)	WRR-1C (%)	TRR (%)
DBN-AE-BLSTM	**90.73**	59.45	81.91	39.39
MLP-AE-BLSTM	88.50	59.95	79.02	33.19
ConvNet-BLSTM	**94.36**	**71.26**	**86.77**	**55.03**

that the *DBN-AE-BLSTM* slightly outperforms *MLP-AE-BLSTM* by almost 2 points in terms of CRR. This is due to the fact that RBMs are pre-trained to explore high-level character structures which are then projected to the features space.

In addition, more than 71% of words are correctly recognized with *ConvNet-BLSTM* and this rate increases by 16 points if we consider the *WRR-1C* measure, i.e. 16% of the recognized words differs by only one character from the ground-truth. This suggests that a post-processing step based on a dictionary, for instance, can significantly improve the word recognition rate.

Tests have also been conducted on **ALIF_Test2** and results are presented in Table 5.4. We can notice the slight degradation of performance for the three methods. **ALIF_Test2** contains a wider diversity of text images than **ALIF_Test1**.

TABLE 5.4: Recognition results on **ALIF_Test2**.

	CRR (%)	WRR (%)	TRR (%)
ConvNet-BLSTM	90.71	65.67	44.90
DBN-BLSTM	87.64	52.33	31.54

5.5.2.1 Supervied vs. unsupervised feature learning

For unsupervised feature learning, we use the RBMs applied for the DBN AE pre-training and we consider the outputs of the 4^{th} RBM as feature vector of each sliding window. The resulting training features are used to train the BLSTM network as performed for the previous models. The obtained recognizer is evaluated on *ALIF_Test1* and the results are presented in Table 5.5. Although the unsupervised features lead to low recognition results compared to other supervised models, the obtained rates are still surprising compared to the amount of prior information used by the RBMs for training. With just character images and a layer-wise exploration of the data, the model succeeds in extracting relatively relevant features that can be used by the BLSTM. Through these features, the network is able to infer some temporal classification rules without reaching a divergence point. We note that the training have been stopped due to a BLSTM over-fitting and not due to a divergence point. This point opens many perspectives for the unsupervised feature learning if we further improve the training dataset for both this model and the BLSTM one.

TABLE 5.5: Suervised vs. unsupervised feature learning. Evaluation performed on **ALIF_Test1**.

	CRR (%)	WRR (%)
Unsupervised-BLSTM	81.37	41.38
ConvNet-BLSTM	94.36	71.26
DBN-BLSTM	90.73	59.45
MLP-BLSTM	88.50	59.95

5.5.2.2 Impact of the number of LSTM cells

We have also studied the impact of the number of LSTM cells per hidden layer on the recognition performance. For this, several BLSTM architectures have been considered in the *DBN-AE-BLSTM* method. The obtained results, shown in Figure 5.16, indicate that a maximum *CRR* is reached with 300 or 350 cells, with no significant improvement with more cells. Thus, all of our experiments have been carried out using BLSTMs with 300 hidden cells in both directions. The number of 300 achieves the best trade-off between the recognition performance and computation complexity.

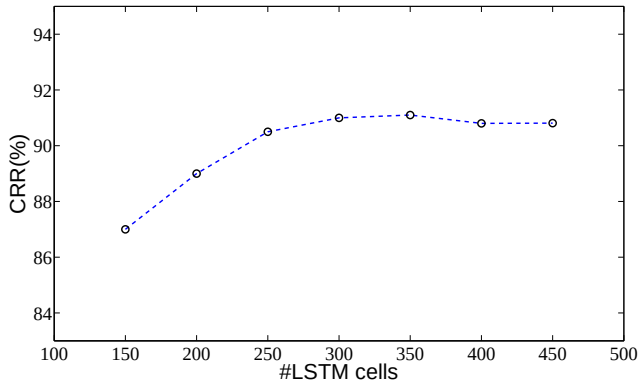


FIGURE 5.16: Impact of the number of LSTM cells.

5.5.2.3 Connectionist vs. non-connectionist features

Comparison to Hand-crafted features

In order to evaluate the advantage of learned features, we have compared our methods to hand-crafted features-based approaches like the one proposed in [GLF+09]. Hand-crafted features are computed as follows: a binarization step is applied on text images. A GMM classifier is trained to separate text and background classes. As in [GLF+09], 9 geometrical features are extracted per column from the resulting binary text images. A BLSTM-CTC has been trained using the obtained feature sequences. The resulting method is called *HC-BLSTM*. The comparative results are illustrated in Table 5.6. The *HC-BLSTM* method is outperformed by the *ConvNet-BLSTM* one by almost 19%. As previously explained, learned features are built under the fixed goal of a better character

representation. They give more valuable information to BLSTM-CTC and cope with background complexity more than hand-crafted features.

Comparison to HOG features

These descriptors are also compared to histogram of oriented edges (HOG) features [DT05]. The proposed text feature extractor is illustrated in Figure 5.17. We consider normalized text images of fixed height $h = 36$ pixels. We sweep, vertically and horizontally, each image using squared windows of 24×24 pixels. The distance between window centers is of $\frac{h}{6}$ pixels. HOG features are densely extracted on a regular grid manner. We consider 8-by-8 squared blocks and concatenate 2-by-2 cells to obtain a descriptor at each grid location. Each horizontal sweeping step $\frac{h}{6}$ of the text image is considered as a time-step where 6 vertically superposed windows are described by HOG features.

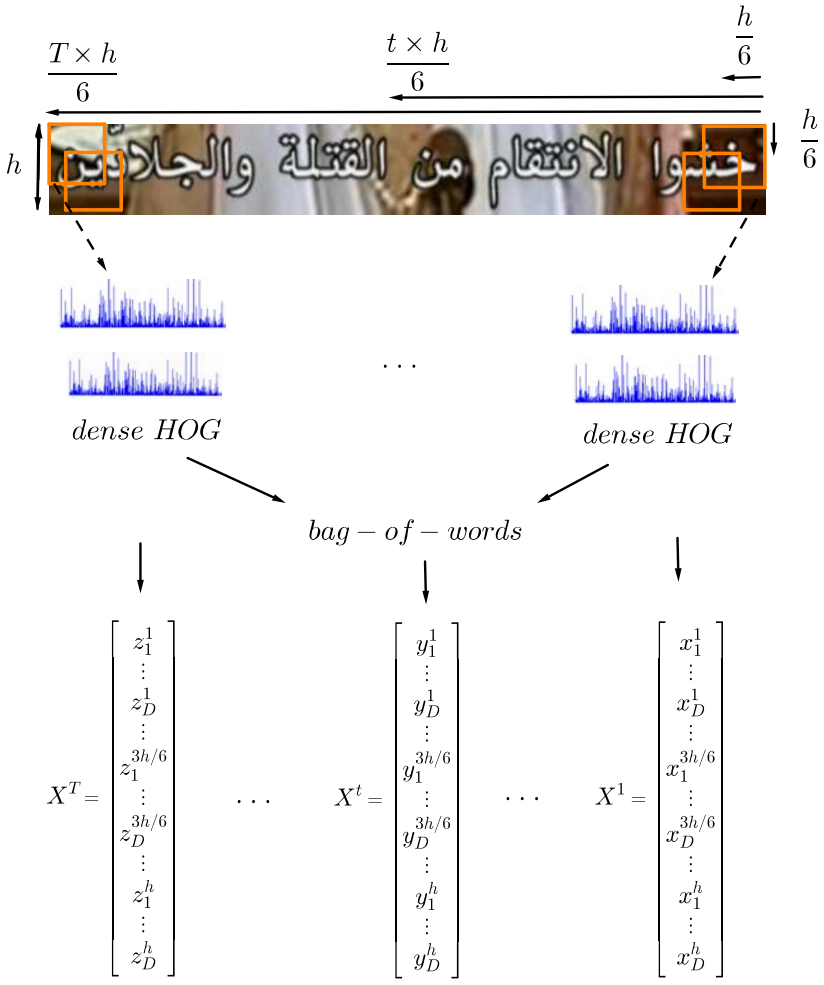


FIGURE 5.17: HOG-based text feature extraction.

Separately, we apply the same feature extraction process on the training character images of *ArabCharSet* and we learn a dictionary of 80 visual words using k-means [Elk03] on a random sampling of the extracted features. The bag-of-words paradigm, described in [WYY+10], is then applied in order to encode the obtained patches of each text image to dictionary entries.

Each resulting feature vectors obtained at each time-step are concatenated to form a global feature vector of $480 = 80 \times 6$ values each. We apply this projection procedure on the *ALIF* dataset in order to describe training and test text images by sequences of HOG-based features. The training sequences are used to fed the BLSTM-CTC network and learn sequential labeling. We use the same parameter settings for the BLSTM as this previously described. Once trained, we evaluate the system, denoted HOG-BLSTM, on *ALIF_Test1*. The obtained results are illustrated in Table 5.6 and show low recognition rates compared to other methods. We note here that we tried to train BLSTM-CTC network on pure HOG features without using the bag-of-words and the training failed to converge. However, using the projection-based features, the system gets to learn temporal classification. Nevertheless, its discrimination ability is still under the ConvNet-BLSTM method by almost 15% in terms of CRR.

Comparison to commercial OCR system
We have also performed a comparative study of the performance of our method w.r.t. an off-the-shelf OCR solution. We have chosen a well-known OCR engine, *'ABBYY Fine Reader 12'*[1]. The Arabic OCR component of this engine has been applied on *ALIF_Test1*. Results have been evaluated using *CRR* and *WRR*. Our *ConvNet-BLSTM* method (and also the 2 other proposed methods) still outperforms the commercial solution with almost 11 points in terms of *CRR* (*cf.* Table 5.6).

TABLE 5.6: Comparative study on *ALIF_Test1*.

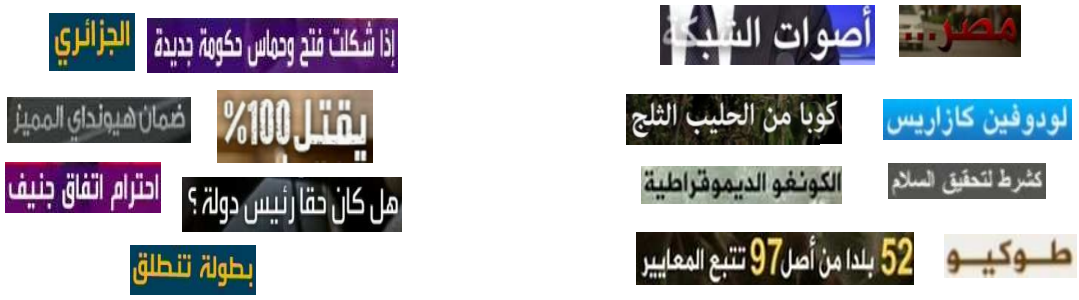
	CRR (%)	WRR (%)
ConvNet-BLSTM	**94.36**	**71.26**
HC-BLSTM	85.44	52.13
HOG-BLSTM	79.47	32.97
ABBYY	83.26	49.80

5.5.2.4 Problem of generalization

At the beginning of training the BLSTM-CTC network, we used a larger dataset that includes, in addition to the *ALIF* examples, a set of text images with one font[2] extremely different from other fonts. Some examples that illustrates this heterogeneity are presented in Figure 5.18.

The evolution of the training in terms of CRR on the training and validation sets is reported in Figure 5.19 using the DBN-AE-BLSTM schema. A comparison between the two curves reveals a particular problem of generalization. For the few first training iterations, the CRR evaluated on the validation set is close to the CRR of the training set. However, beyond almost the 15^{th} iteration, the CRR of the training set begins to exceed validation rates by almost 10%. This gap continues to increase throughout training to reach almost 15% at the 39^{th} iteration (cf. Figure 5.19). The CRRs continue even to increase beyond these iterations to reach values close to 99% while CRR on the validation set remains in the range of 75% which refers to an over-fitting situation.

[1] http://finereader.abbyy.com/professional/
[2] We refer to this font as the 'extra' font.



(a) *The extra font* (b) *Other fonts*

FIGURE 5.18: Illustration of the extra font.

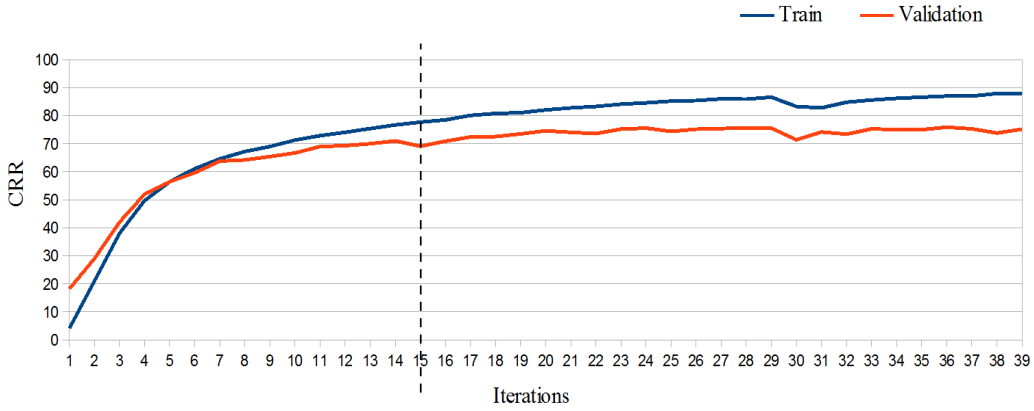


FIGURE 5.19: Evolution of initial training (the considered extra font is included in training set).

Among solutions that we have tried, we have eliminated this extra font from the whole dataset and we present the novel training evolution in Figure 5.20. The obtained training, validation and test sets, which form the *ALIF* dataset and include many channels, remains challenging regarding the large variety in text properties and acquisition conditions even without this font. Despite this, our system succeeds to generalize and reaches high CRR on test sets.

The obtained results in this curves show that this extra font is very critical for training. It represents almost the same properties of *ALIF_Test3* and it is very difficult to find a generalization point in the recognition process if we include it. Examples with this special font clearly need a separate training schema or a larger training set for the BLSTM-CTC network. For the first possible solution, it is essential that the final OCR system includes a prior font classification step in order to choose the proper recognizer. This classification can be optical or based on the channel given that most channels use almost one specific font. For the second solution, it can be viewed as pushing the overfitting point. By increasing the number of training examples, the BLSTM-CTC can learn more data modalities and see more feature combinations through time in order to infer correct characters, paws and even learn directly word patterns.

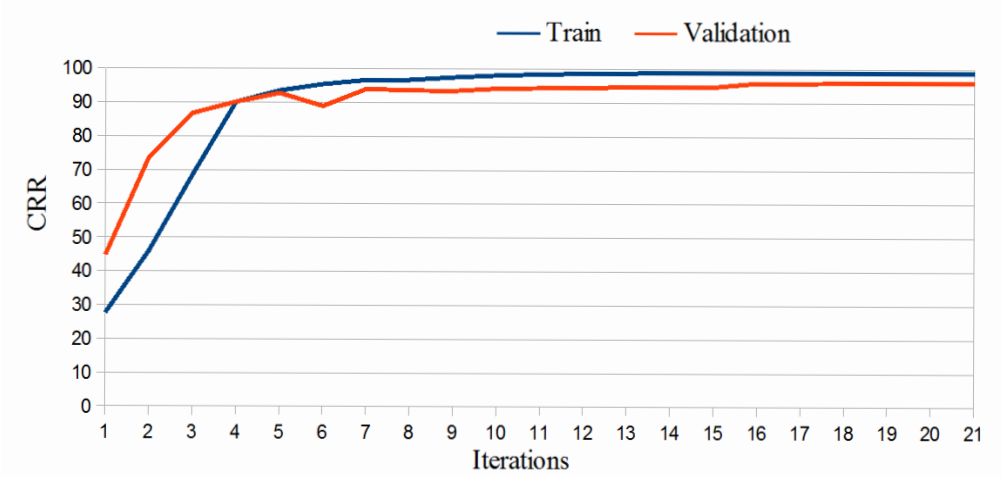


FIGURE 5.20: Evolution of training without the extra font.

At the same time, reaching these recognition rates on the *ALIF* dataset with only this amount of training examples (basically in the range of 4000 text images) reflects the robustness of the proposed methods.

5.6 Conclusion

We have presented in this chapter our Arabic text recognition solutions. The proposed methods are segmentation-free and avoid all problems related to the cursivness of the Arabic script. Based on a BLSTM-CTC network, the OCR systems learn temporal classification of sequences of text features without prior knowledge of text unit boundaries. Using this recurrent connectionist solution with a specific objective function defined by the CTC, the system learns the transcription task without relying on prior parametrization to fit data structure like in HMM-based systems. The BLSTM network is fed with sequences of text features that, hopefully, give relevant representation of the data. In order to overcome all challenges related to text, background and acquisition condition complexity, specially for video contents, we proposed to learn textual features instead of using hand-crafted descriptors. We considered features generated by learned character-based auto-encoders and a ConvNet classifier. These learned models are applied in a multi-scale fashion on the text images in order to transform them into sequences of learned features that fed the BLSTM-CTC system.

The proposed methods have been extensively evaluated on the public *ALIF* dataset and gave high recognition rates. The used features have been compared to other non-connectionist features, namely column- wise hand-crafted features and HOG descriptors an proved to outperform them with large gaps in recognition rates. Our OCRs were also compared to commercial OCR engine and obtained the best results.

Despite their efficiency, the proposed OCR systems still produce some errors that seem obvious to correct using linguistic information. In the next chapter, we will address this issue in an attempt to further boost recognition results using language models. We propose to deal with connectionist and frequency-based linguistic models with a specific integration of this information into the decoding schema.

Chapter 6

Language modeling

Contents

6.1 Introduction

In the previous chapter, we have presented the optical recognition schemes that give high recognition rates thanks to the chosen features and the labeling process. However, a look at the false transcriptions shows some of the errors can be efficiently corrected using language information. For a LSTM OCR system that basically yields high recognition rates, introducing proper language models is not a trivial task. It can easily deteriorate results. Indeed, for such OCR system, errors do not necessarily reflect low OCR confidence (in terms of probability outputs) in some locations of the transcription. Therefore, in order to enhance recognition results, one must find the best confabulation between the OCR and the language model components and first of all, the most appropriate method to represent language modalities.

As presented in Chapter 2, for handwritten or printed text, language models are often integrated as a post-processing step in order to correct the recognition errors. They are widely used in the decoding graph of segmentation-based OCR systems or to re-rank best final hypothesis of a HMM-based recognizer. Usually, results are reported when

there is a considerable improvement. This explains partly the fact that language models are rarely used with highly performing OCR systems like those based on LSTM. In some works, the high performance of the LSTM can make thinking that it is useless to introduce additional linguistic information which can be cumbersome for the OCR [UHB13, BUHAAAS13] and can induce some *hallucination* of incorrect words instead of correcting some obvious errors. However, this point of view may be discussed. On one hand, the use of explicit language models in the OCR scheme is not a trivial task. It depends on the sophistication of the model and on the integration paradigm within the whole solution. On the other hand, for an OCR system, the LSTM network is mainly trained with a sequence or temporal classification goal. It can take into account long range inter-dependencies in the input sequence which may store some linguistic information. However, the main goal of the training phase is to classify the current *optical* observation and not to predict the next character or word. Thus, this implicit linguistic information incorporated in a LSTM network may not be relevant enough. It depends on how much chosen text features are accurate to represent characters and words and also on the accuracy of the alignment-based objective function. Besides, for an additional learning of the language modalities, the network needs huge amount of training data. Therefore, making use of explicit linguistic information in parallel with the LSTM classification outputs should enhance the final results.

In this chapter, we focus on two main factors to reach better improvements. First, we propose to take advantage from the latest advances in language modeling, namely the connectionist models. We propose to learn Arabic language modalities using Recurrent Neural Networks that are able to capture long range linguistic dependencies. We use simple RNN models and models that are learned jointly with a Maximum Entropy language model and compare them to frequency-based models. Second, for the decoding schema, we are not limited to a n-best re-scoring of the OCR hypotheses. Instead, we propose a modified beam search algorithm that uses both OCR and language model probabilities in parallel at each decoding time-step. We introduce a set of hyper-parameters to the algorithm in order to boost recognition results and to control the decoding time.

Different learning architectures and parameters are explored. The whole schema is exhaustively evaluated on the *ALIF* dataset. Thanks to the decoding algorithm and the integration of the RNN-based LM, we reach an improvement of almost 16% over the baseline BLSTM-based OCR system in terms of WRR. Compared to frequency-based LM, namely n-grams, the proposed connectionist LMs achieve best results both in terms of entropy and WRR after joint decoding.

In the reminder of this chapter, we first present the proposed language models in Section 6.2. Section 6.3 is dedicated to the joint decoding process using both language model and OCR responses in addition to the introduced parameters to control the decoding accuracy and speed. Section 6.4 describes all the experimental set-up of the used language models, namely the training process and the different used architectures. We present also in this section the empirical tuning the decoding parameters and we provide and discuss different improved recognition results.

6.2 Language modeling

In statistical language modeling, the most used atomic units are words given their accuracy compared to character-level LMs. However, one of the most important drawbacks

of word-level LMs is their incapacity to completely handle new words, known as Out-of-Vocabulary (OOV) words. Many methods have been proposed to alleviate this problem by training models to sub-word level [PDSR11] or handling only OOV words with character-level LM [Szö10]. However, for some rich languages like Arabic, the OOV rate remains very high specially when we consider unconstrained language modeling. Another problem which may face word-level LMs in Arabic language is the diacritization. Indeed, for Arabic, we can find two or more words having the same letters but when diacritized they have different meanings and pronunciations. They can have even different grammatical classes between verbs, nouns, adjectives, etc. This strongly influences the accuracy of the word-level LM since most of Arabic corpora are not diacritized which partly explains the difference of perplexity levels between Arabic and other languages LMs. Even some words can, semantically, incorporate many other words. For example the word 'أعطاهم' means 'He gave them'. It is the concatenation, by a grammar rule, of two Arabic words: the word 'أعطى' which means 'He gave' and the word 'لهم' which means 'to them'. Therefore, building a proper Arabic word-level LM requires huge amount of data in order to first define a proper Arabic vocabulary and to explore all these language modalities and varieties.

For these reasons, we consider in these work only Arabic character-level LMs to deal with unconstrained Arabic text Recognition. We are not limited to a certain lexicon nor to frequency-based LMs. Our LMs are connexionist and based on RNNs in order to capture long range language dependencies. We use simple RNN models and models that are learned jointly with a Maximum Entropy model.

6.2.1 RNN-based language modeling

As presented in Section 2.5, first connexionist LMs have been proposed more than a decade ago with feed forward neural networks LMs introduced by Elman [Elm90] and Bengio [BDVJ03]. The core idea was to avoid exact presentation of the context and find instead a projection-based representation of it in a learned lower dimensional space. The method was successfully applied for unconstrained language modeling specially in large-scale corpora [SG02, ZMFEB+14]. However, for these models like also n-gram LMs, the history is still limited and pre-fixed to a certain range (few words). RNN-based LMs were proposed mainly as a solution to this problem among others [KMKB11, MDK+11, MKB+10, MZ12, Mik12].

Like other neural networks, RNNs as 'fuzzy' models do not directly use exact templates of training examples which is the case for n-gram models. For these latter, predictions are based on counting exact matches to the pre-stored templates during training. Connectionist LMs, instead, try to explore data and its internal states perform high-dimensional interpolation between training examples.

In particular, for RNNs, the history is not limited to a fixed range. Given the recurrent connections, the network is able to handle long range context patterns and considers, theoretically, all the history. Its recurrent internal states allow to learn useful representation of the context from data. Besides, compared to neural network LMs, RNN models are less expensive in terms of number of parameters and training data [MZ12]. Thanks to the recurrent architecture, the RNN do not have to use specific parameters to encode unit positions in the history like in the case of feedforward neural networks.

Instead, this information can survive in the hidden recurrent states of the network and hence can be used later. Its is a specific complex encoding of the history.

In this work, we use the RNN LM architecture proposed in [MKB+10]. It is presented in Figure. 6.1. The network is trained at character level. The text is presented as sequences of individual characters $(c^0, \cdots, c^t, \cdots, c^T)$ belonging to the Arabic alphabet Σ. Thus, the size V of the vocabulary is equal to $|\Sigma|$. The input layer is a concatenation of two vectors. The first vector C^t represents the current character c^t encoded as $1 - of - V$ vector where only the value corresponding to the index of the carrent character is set to 1 and others to 0. The second vector represents the state of the hidden layer at the previous time-step s^{t-1}. This recurrent connection allows the network to model the language inter-dependencies. The hidden units with sigmoid activations are connected to a softmax output layer o^t of size V. The latter represents the probability distribution of the next character c^{t+1} given his context $P(c^{t+1} \mid c^t, s^{t-1})$. The weight matrices W_{ch}, W_{hh} and W_{ho} represent the different connections between the network layers (W_{ch} and W_{hh} between input and hidden layers and W_{ho} between hidden and output layers).

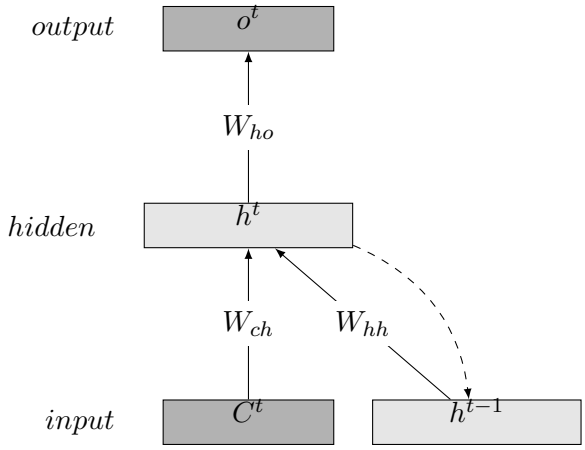


FIGURE 6.1: RNN-based LM architecture

The output of a hidden unit j at time-step t is expressed as follow:

$$h_j^t = \sigma\left(\sum_i (W_{ch})_{ij}^t c_i^t + \sum_k (W_{hh})_{kj}^t h_k^{t-1}\right) \tag{6.1}$$

where σ is the *sigmoid* activation function. As for the outputs of network, they are calculated as follow:

$$o_j^t = softmax\left(\sum_i (W_{ho})_{ij}^t h_i^t\right) \tag{6.2}$$

Where $softmax$ is the softmax function that ensures a probability distribution over characters. The network is trained using truncated BPTT, described in [RHW86], to avoid two extreme cases during training: propagating gradient errors at each time-step or after processing the whole training data. The first case leads to high computational complexity and does not give the network the opportunity to store relevant context information that can be useful in the future. The second case is the bottleneck of the *vanishing gradient* problem explained in the previous chapter. So, instead, the network

is unfolded in time for a specified amount of time-steps. Moreover the update of weights is done in mini-batch mode (after processing several training examples).

6.2.2 RNNME: Joint learning of RNN and ME LMs

Given a character c, its history h and a set of features f weighted by λ, the Maximum Entropy (ME) [Ros94, BPP96] model can be expressed as follow:

$$P(c \mid h) = \frac{e^{\sum_{i=1}^{N} \lambda_i f_i(h,c)}}{\sum_c e^{\sum_{i=1}^{N} \lambda_i f_i(h,c)}}, \tag{6.3}$$

Usually, features are hand-designed. They can be n-grams or any integrated linguistic or syntactic information and constraints built from these features. Training the ME model consists in learning the set of weights λ. The model with the greatest entropy consistent with the constraints is the same as the exponential model which best predictive capabilities. ME models show state-of-the-art performances on broadcast speech recognition with a specific shrinking model, called model 'M'[Che09, CMR+09]. This ME model proved high performances for Arabic ASR in the context of DARPA's Global Autonomous Language Exploitation (GALE) program. It largely outperforms a baseline LM that consists on a linear interpolation of in-domain trained 4-gram models (with modified Kneser-Ney smoothing) [CMR+09], which is relatively strong as a baseline in the context of ASR.

In [Mik12], it has been shown that ME models trained with n-gram features have similar performance as usual backoff models with modified Kneser-Ney smoothing. The ME model has been viewed as a neural network with no hidden layer. It consists of a weighted matrix 'W_{ME}' that directly connects the input layer that represents features to the output layer. Figure 6.2 illustrates a simple example of ME model with trigram features drawn from a vocabulary of 3 Arabic characters (Siin 'س', Nuun 'ن' and Yaa 'ي'). This graphical representation shows all the bi-gram history combinations taken as inputs and vocabulary units at the output level. For n-gram features, such representation leads to a very large set of parameters equal to V^n corresponding to the different vocabulary combinations. Besides, a large part of these combinations will occur just few times in he training set. This memory waste and complexity can be alleviated by using a hash function that maps each n-gram history to a single value in a smaller hash table.

In the same work, the authors propose to train the ME model jointly with the RNN model using stochastic gradient descent. In this resulting model called RNNME, direct connections are added between the RNN output layer and the sparsely coded input layer. Theoretically, using this joint learning can improve performance with a hidden layer of small size. Indeed, learning a RNN LM with a small vocabulary, such in our case with character level LM, requires a large number of hidden units to cover all the history patterns. Using the RNNME model, the RNN learning will just focus on complementary information to the direct binary connections. Basic patterns can be hence described by n-gram features of the hash-based ME model. The model is able to keep sufficient results with less hidden units decreasing hence the training complexity. However, this model still requires a large memory for the hash representation. So depending on the application, a tradeoff between speed, performance and memory has to be found.

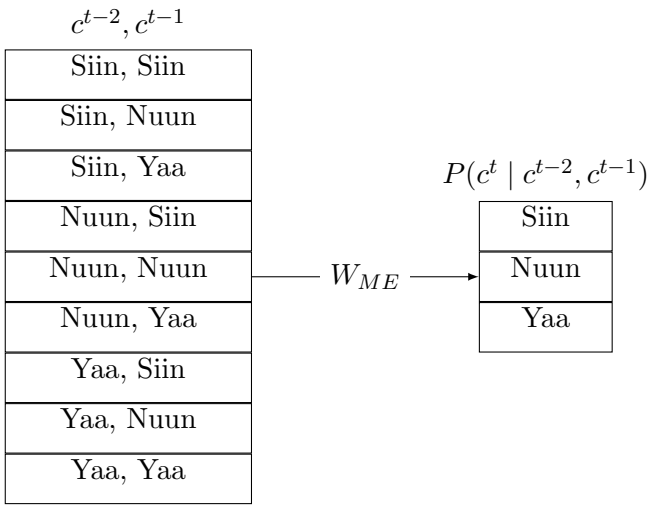


FIGURE 6.2: Maximum Entropy model with 3-gram features.

Although this frequency aspect of n-grams adds a kind of simplicity to the LM, it makes handling long range linguistic dependencies more difficult. Indeed, n-gram probabilities can be stored in precomputed tables which ensures a high speed. However, for massive amount of training data, a large part of patterns cannot be effectively represented and discovered during training since that the number of possible n-grams increases exponentially with the length of the context. This problem with n-grams presents the most important drawback of these LMs that the proposed neural network-based LM tried to fix in many occasions (cf. Section 6.1 and Section 2.5).

6.2.3 N-grams

One of the most used statistical LMs is n-grams. Known as frequency-based LMs, n-grams estimate probability of an upcoming granularity c (character in our case) given its history h as follow:

$$P(c \mid h) = \frac{count(hc)}{count(h)}, \tag{6.4}$$

where $count(hc)$ refers to number of times the sequence of hc characters appears in the training set. Using the chain rule, the probability of a sequence of characters $C = (c_1, \cdots, c_i, \cdots, c_N)$ of length N is expressed as follow:

$$P(C) = \sum_{i=1}^{N} P(c_i \mid c_1, \cdots, c_{i-1}) \tag{6.5}$$

The order n of the n-gram LM is defined by the length of the context h. For instance, a tri-gram LM have a context range of 2 and for a 5-gram, $|h| = 4$. This order is a very crucial parameter that influences largely the LM performances. There is no rule for the choice of the better order, but this still depends on the application, namely the used granularity (words or characters), the language , the size of the vocabulary and the training set, etc. Usually its is empirically set-up and larger orders do not necessity reflect better performances.

Another crucial point that influences the accuracy on n-grams is the smoothing technique. During the estimation of the conditional probability expressed in Equation 6.4, the training set may not contain a character c in a specific context h. A probability of zero is going to be assigned to these cases. In order to alleviate this sparsity in the estimated probabilities' vector, a smoothing operation must be applied. It consists in redistributing probabilities between frequent and non-frequent n-grams. Many smoothing techniques have been proposed in the literature like the Kneser-Ney smoothing [KN95], Good-Turing smoothing [Kat87], the Witten–Bell discounting [WB91], etc. An empirical study of smoothing techniques has been tackled in [CG99].

6.3 Decoding schema

At this level, given a text image presented as a sequence of features X, our BLSTM-based OCR system gives as output a sequence of probability vectors $(y_0, \cdots, y_t, \cdots, y_T)$. At each time-step t, the vector y_t provides the probability distribution over character classes including the BLANK class. The decoding stage aims to find the most probable transcription given these outputs (given the input text image in other terms). This can be pretty simple if the outputs present peaks corresponding exactly to the target labels and in the same order. Decoding in this case can be performed by the Best Path algorithm that consists in taking only maximum responses at each time-step. However, in real applications, this is not the case.

In this context, the Beam Search (BS) algorithm is more accurate for decoding real OCR outputs. The core idea of the algorithm is to explore the graph defined by BLSTM outputs at each time-step and for every character. At each step, the algorithm proposes to be extended a given number of the most likely hypotheses. The goal is not to choose the character with the highest response but instead, a set of extensions that maximize the most the likelihood of the whole actual path. The number of considered hypotheses at each time-step defines the beam width W of the algorithm.

A crucial point about the BS algorithm is that it allows the integration of LM information during decoding. It is also able to deal with both multiple segmentations and multiple hypotheses per window. Inspired by the work of [GJ14],we propose a modified version of the BS algorithm that directly performs joint decoding using only the LM and the LSTM responses. In [GJ14], the authors combined LM and acoustic-based recognition in order to re-score the n-best list from a set of candidate transcriptions produced by a HMM speech recognizer. In our case, we start from an empty space of hypotheses without any re-scoring of existing set of hypotheses and we expand it based only on LM and LSTM responses. This decoding schema allows to treat free-segmentation OCR problem as a direct sequence transcription process without relying on any pre-definition of model structures like in the case of HMM-based systems [SZGH09, RRS+13, EBCBGMZM11].

In this algorithm, described in Algorithm. 1, both the network outputs and LM responses are interpreted as transition probabilities (in HMM terms). The goal is to find the transcription sequence S that maximizes the flowing probability:

$$p_{OCR}(S \mid X)(p_{LM}(S))^{\alpha} \tag{6.6}$$

where p_{OCR} refers to the probability based on the OCR outputs and p_{LM} is the prior probability weighted by an α factor. This factor is used to weight the impact of the LM during decoding.

To reach this goal, the algorithm defines two sets of prefixes at each time-step: $Prefix$ and Hyp. The set $Prefix$ contains the hypotheses or prefixes retained at the previous time-step. The set Hyp contains hypotheses that are proposed to extend at the next time-step. The total probability of a prefix sequence S (partial output transcription) at a given step t is defined as follow:

$$p(S,t) = p_{nb}(S,t) + p_b(S,t) \tag{6.7}$$

where $p_{nb}(S,t)$ and $p_b(S,t)$ refer to *BLANK* and *non-BLANK* probabilities. In other terms, these are the probabilities of the prefix S ending with *BLANK* or not ending with *BLANK* given the first t time-steps of the input X. Indeed, making a distinction between the two probabilities is essential specially when removing label repetitions on successive time-steps.

The probability of extending the prefix S by a character c at the time-step t is defined by:

$$p_{ext}(c,S,t) = y_t(c)(p_{LM}(c \mid S))^{\alpha} \begin{cases} p_b(S,t-1) \text{ if } S_{|S|} = c \\ p(S,t-1) \text{ otherwise} \end{cases} \tag{6.8}$$

where $y^t(c)$ is the OCR output for the character c at the time-step t, $p_{LM}(c \mid S)$ is the response of the LM (probability of getting c given the history S obtained from the output layer of the RNN and RNNME LMs) and $S_{|S|}$ is the last label of the sequence S.

Intuitively, when extending a sequence S by a character c, the impact of the LM probability becomes more relevant when the context is long enough. Hence, instead of using a unique LM weight, we define a dynamic setting of α. Empirically, we increase the LM strength when reaching a context length of 3. Thus, α is defined as follow:

$$\alpha = \begin{cases} \omega_1 \text{ if } |S| \geq 3 \\ \omega_2 \text{ otherwise} \end{cases} \tag{6.9}$$

Given these probability definitions, we are able to integrate the linguistic information in parallel with the OCR system responses. Usually, existing works that use the BS algorithm are often limited to LM-based rescoring of the W best sequence hypotheses that are already selected based only on recognition probabilities. Here, Algorithm 1 makes possible the integration of the LM outputs with the BLSTM responses in order to compute all the extension proposition probabilities from the beginning of the decoding process and at every step. Hence, the score of a hypothesis at any time-step of the algorithm will depend on visual and linguistic priors. Moreover, we propose to use the two following pruning criteria in order to eliminate potential wrong hypotheses from the set Hyp at each time-step during decoding:

$$hyp1(S,t) = \{S \in Hyp \mid rank(S,t) \leq th_{rk}\} \tag{6.10}$$

$$hyp2(S,t) = \{S \in Hyp \mid p(S,t) - p(S_{best},t) \leq th_{sc}\} \tag{6.11}$$

Algorithm 1 Sequence decoding

$Hyp \leftarrow \varnothing$
$p_b(\varnothing, -1) \leftarrow 1$
for $t = 0$ **to** $T - 1$ **do**
  $Hyp \leftarrow \{\}$
  for $S \in Prefix$ **do**
    if $S \neq \varnothing$ **then**
      $p_{nb}(S, t) \leftarrow p_{nb}(S, t-1) y^t(S_{|S|})$
      if $S_{1:|S|-1} \in Prefix$ **then**
        $p_{nb}(S, t) \leftarrow p_{nb}(S, t) + p_{ext}(S_{|S|}, S_{1:|S|-1}, t)$
      end if
    end if
    $p_b(S, t) \leftarrow y^t(BLANK)(p_{nb}(S, t-1) + p_b(S, t-1))$
    $p(S, t) = p_{nb}(S, t) + p_b(S, t)$
    Add S to Hyp
    for $c \in \Sigma$ **do**
      $S\_ext \leftarrow concat(S, c)$
      $p_b(S\_ext, t) \leftarrow 0$
      $p_{nb}(S\_ext, t) \leftarrow p_{ext}(c, S, t)$
      $p(S, t) = p_{nb}(S, t) + p_b(S, t)$
      Add S_ext to Hyp
    end for
  end for
  $S_{best} \leftarrow \underset{S \in Hyp}{\operatorname{argmax}}\, p(S, t)$
  $Prefix \leftarrow \{S \in Hyp \mid rank(S, t) \leq th_{rk}\} \bigcap \{S \in Hyp \mid p(S, t) - p(S_{best}, t) \leq th_{sc}\}$
end for
return $\underset{S \in Prefix}{\operatorname{argmax}} \left(p(S, t)^{\frac{1}{|S|}}\right)$

Criterion (6.10) defines a fixed number $th_{rk} = W$ of hypotheses to retain based on their rank among all the possible hypotheses. At a time-step t, we retain just the first best th_{rk} prefixes S in terms of $p(S, t)$ (cf. Equation (6.7)). Using only this pruning rule, excluding hypotheses with very low probabilities is not always guaranteed. The decoding process may remain therefore very expensive. Thus, criterion (6.11) is used to limit dispersion in hypotheses scores by a threshold th_{sc}. In principal, this can reduce computation time and increase decoding performance. It can prevent very weak prefixes from a future extension to a very probable sequence which is non other than a LM *hallucination*.

6.4 Experimental setup, results and discussion

For the BLSTM network training, all Arabic letter shapes have been considered. The OCR component here considers the different shapes of a letter depending on its position in a word. It is based on the ConvNet-BLSTM schema presented in the previous chapter. However, in this work, for Arabic language modeling and final text transcription we consider atomic Arabic letters. Thus, we map recognized labels into 54 letters of the Arabic keyboard letters. In this work, our goal is to study the contribution of the

language modeling in optical text recognition in parallel with the effect of their integration paradigm into the decoding stage. Therefore, results are reported in terms of Word Recognition Rates (WRR). We take into account the final 54 letters-based transcription and we use the 'SPACE' character to segment the character outputs into words. We note that for the final evaluation, we do not consider examples with punctuation marks and Arabic digits. These special characters are not well presented in the *ALIF* dataset. Taking them into account can cause an unfair integration of LM responses in the decoding stage, regarding the case of other letters.

6.4.1 Language models set-up

To train the different LMs, we built a dataset of Arabic text lines from 3 main sources:

1. Ajdir Corpora[1]
2. Watan-2004 Corpus/Khaleej-2004[2] [ASB11]
3. Open Source Arabic Corpora[3] (OSAC) [SA10]

Texts of the first two sources have been extracted from Arabic newspapers. The third source includes texts collected from Arabic TV channels websites like BBC and CNN. Initially, the gathered texts contain lines with different lengths (paragraphs, single words, short and long sentences). In general, embedded texts in TV broadcast (the main applicative concern of this paper) are often limited to few words per line. Thus, to fit this context, we cut the obtained text (dedicated to LM) into text lines with a limited number of words. Moreover, we apply some other pre-processing steps like removing non-Arabic characters, digits and extra punctuation marks. We remove also an important number of text repetitions. The words are then split into individual characters and the space between words is replaced by a specific label.

The obtained dataset contains in total 52.08M characters. We randomly select a subset of text lines with 44.47M characters to train the LMs. The remaining text lines are split into two subsets: one with 4.29M characters for validation and the other, denoted ***TEXT_Test*** set, with 3.32M characters for test.

We train different RNN and RNN-ME models using the stochastic gradient descent. An evaluation of the entropy on the validation set is carried out at each training epoch in order to keep the best network weight configuration. We set the learning rate to 0.1. It is divided by a factor of 1.5 at each epoch if the validation entropy increases. The value of unfolding steps is fixed to 8. This allows the network to learn information storage for more than 8 time-steps in a reasonable training time. A batch size of 20 is used for gradients averaging. We use different sizes of the hidden layer in order to study the effect of this parameter and choose the optimal one. The training process is stopped when almost a constant validation entropy is reached.

We train, in parallel, n-gram LMs using the SRILM toolkit [Sto02]. The models are smoothed using the Witten-Bell discounting and the order is tuned on the validation

[1] http://aracorpus.e3rab.com/argistestsrv.nmsu.edu/AraCorpus/
[2] https://sites.google.com/site/mouradabbas9/corpora
[3] https://sites.google.com/site/motazsite/arabic/osac

set. Best entropy results are obtained with a 7-gram LM with no significant improvement beyond this order.

A first level evaluation of the LMs is carried out with respect to the character entropy criterion on the *TEXT_Test* set. This gives an idea about the performance range of the learned LMs independently from the application. However, as previously mentioned, our final goal is to analyze the contribution of language modeling in Arabic text recognition in TV Broadcast. Therefore, in addition to the entropy, we apply the whole recognition schema, including OCR and LM responses, on the text images and we evaluate results in terms of WRR. This allows to discover the range of correlation between the LMs entropy and their contribution in terms of WRR enhancement. Indeed, in the problems of joint sequence and LM decoding, its is important to study this correlation for a proper judgment of the LM choice. As for the tuning of the decoding algorithm parameters, it is carried out with respect to the WRR criterion.

For the tuning of the joint decoding schema, we use a separate development set, denoted ***DEV_Set***, that is made up of 1225 text images. To test our final OCR systems, we use text images from ***ALIF_Test1*** and ***ALIF_Test2*** that include texts with only Arabic letters. Examples with digits and punctuation marks have been excluded. The obtained subsets contain 827 and 1175 text images respectively.

6.4.2 Primary results

Initially, we train different connectionist and frequency-based LMs. Our goal, in the beginning, is to have an idea about the range of language information contribution in text recognition and also a primary comparison between the different models independently from the decoding parameters. For the RNN LM, we have trained four models RNN-100, RNN-300, RNN-500, RNN-700 with 100, 300, 500 and 700 hidden units respectively. Similarly, we have trained two RNN-ME LMs with 100 and 300 hidden units (RNNME-100 and RNNME-300). The order of the direct binary connections of the ME part is set to 5.

For numerical stability and speed, all OCR and LM probabilities are transformed to the logarithmic space. We work thus in the $]-\infty, 0]$ interval instead of the $[0, 1]$ interval. The LSTM outputs have been first decoded with no LM using the Best Path Decoding algorithm (BPD). We simply concatenate labels with maximum OCR responses at each time-step after deleting BLANKs and repetitions. A second decoding schema relies on the proposed BS algorithm with no integration of the linguistic information (BS-No-LM). We only use the probabilities emitted by the OCR component. Finally, we propose to integrate the trained LMs responses as presented in Algorithm 1. Depending on the used LM, these methods are denoted BS-RNN-100, BS-RNN-300, BS-RNN-500, BS-RNN-700, BS-RNNME-100, BS-RNNME-300 and BS-7-gram. We set both LM weights ω_1 and ω_2 to 0.4 and the beam width to 4 hypotheses at each time-step ($th_{rk} = 4$) without taking into account th_{sc}.

Entropy/WRR analysis
At a first level, the different LMs are independently evaluated on the *TEXT_Test* set with respect to the entropy criterion in order to have a primary comparison between the different models. At a second level, we introduced the different LMs in the BS

decoding schema as previously described and we evaluate results in terms of WRR on the *DEV_Set* text images. Results are reported in Figures 6.3 and 6.4 respectively.

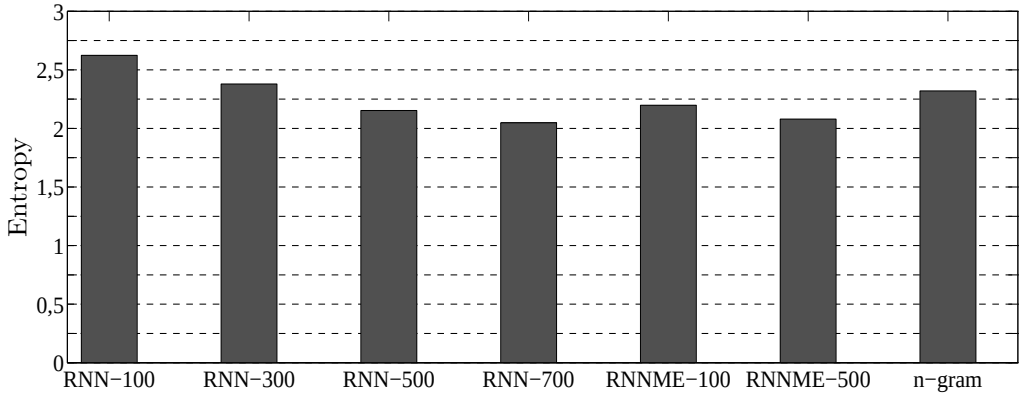


FIGURE 6.3: Evaluation of the proposed LMs in terms of entropy on the *TEXT_Test* set.

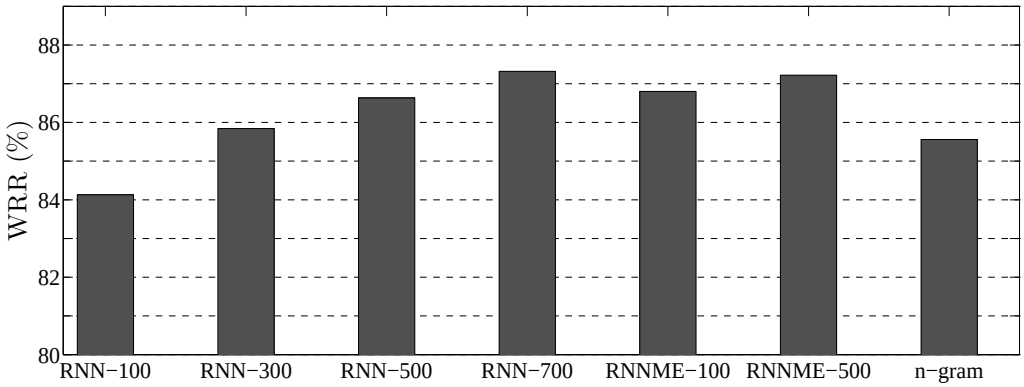


FIGURE 6.4: Primary impact of the proposed LMs integration on the text recognition results - Evaluation on the ***DEV_Set*** text images.

In general, when evaluating different language modeling approaches, the common used criterion is the entropy (or perplexity) that reflects how well the LM as a probabilistic model predicts samples (characters in our case). Lower entropy corresponds to more efficient models. Evaluating this measurement is also extremely important when tackling the problem of joint sequence and LM decoding in order to ensure fair comparison between different final recognition schemas. Results presented in Figures 6.3 and 6.4 show a strong correlation between LMs entropy and final WRRs of the different methods applied to text images. Models with lower entropy yield to better recognition rates when integrated into the decoding. In particular, results show that connectionist LMs outperform n-gram LM both in terms of entropy and WRR. We will further analyze the comparison of the the different models in the next experiments. The most important observation behind these results is that the decoding schema conserves the same performance order of the proposed LMs as this obtained in terms of entropy. This reflects a large part of the correctness of linguistic information integration into the recognition

schema and ensures a primary solid foundation for further improvements of the OCR results.

An additional study of this correlation has been conducted on a single language model. While training the RNN-700 LM, we save different configurations of the model at different training epochs and evaluate their resulting entropy per character on the *TEXT_Test* set. In parallel, we integrate each of these configurations, separately, in the decoding schema and we evaluate their resulting contributions in terms of WRR on the *DEV_Set* text images. Obtained results are illustrated in Figure 6.5 which clearly show progressive entropy reduction during the LM learning. This reduction corresponds completely to a progressive improvement in recognition performances while applying the joint decoding at each of these epochs on the development set text images. This proves once again the correctness of our joint decoding schema that ensures best use of the LM.

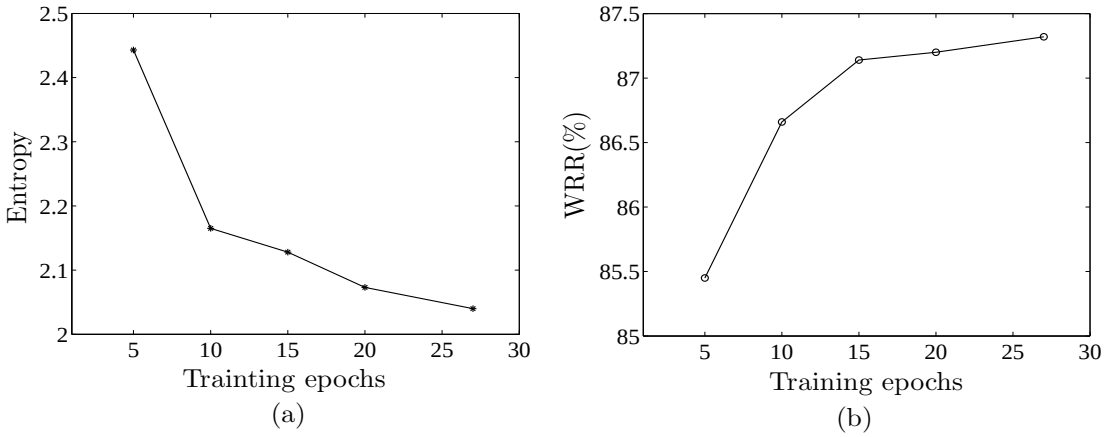


FIGURE 6.5: Evolution of the RNN-700 LM results during training: (a) progressive reduction in the entropy per character on the *TEXT_Test* set. (b) a corresponding improvement in WRR after joint decoding applied to the *DEV_Set* text images, using the RNN-700 configuration after each training epoch.

Primary comparison

In order to further compare the contribution of the proposed trained LMs and the impact of the chosen decoding schema on the recognition results, we evaluate the different methods on the development and test image sets in terms of WRR. Results are illustrated in Table 6.1. Table 6.1 shows very close results of BPD and BS-No-LM schema with a small improvement over the second one. However, when introducing the language information into the decoding algorithm, we obtain considerable improvements in recognition rates (more than 7% for all models compared to the BPD and BS-No-LM baselines). These improvements depend on the used model. Nevertheless, better performances are obtained with the connectionist LMs compared to the n-gram model. For example, on *ALIF_test1*, both RNN and RNNME models allow a maximum gain of almost 13% in terms of WRR compared to the BS-No-LM method. They outperform, therefore, the 7-gram model by more than 2 points.

A look at the RNN LMs results reveals an important impact of the hidden layer size on recognition rates (cf. Table 6.1) and also on the models entropy (cf. Figure 6.3). The WRR increases each time we use more hidden units. As previously mentioned, training

TABLE 6.1: Primary results: performance of the proposed models in terms of WRR on the development and test image sets.

Model	Dataset		
	DEV_Set	*ALIF_test1*	*ALIF_test2*
BPD	77.11%	73.07%	67.79%
BS-No-LM	77.17%	73.53%	68.22%
BS-RNN-100	84.13%	82.46%	77.13%
BS-RNN-300	85.84%	84.98%	79.42%
BS-RNN-500	86.63%	86.02%	79.95%
BS-RNN-700	**87.32%**	**86.74%**	**81.24%**
BS-RNNME-100	86.8%	86.02%	80.55%
BS-RNNME-300	**87.22%**	**86.56%**	**81.14%**
BS-7-gram	85.56%	84.11%	78.41%

character-level LMs based on RNNs requires a large hidden layer. With small vocabulary that is limited here to characters as input, using large number of hidden units is essential to maintain a reasonable number of the network parameters for a better learning of the language modalities. As shown in Table 6.1, by using a model with 700 hidden units, we are able to improve WRR results by more than 3 points compared to a 100-hidden layer RNN. Nevertheless, the improvement of this later with a heavy architecture (100 hidden units) is still not negligible compared to the baseline BPD-No-LM (more than 9 points for *ALIF_test2*).

As for the RNNME-based models, the results are more surprising. With much fewer hidden units, these LMs are able to achieve comparable results to RNN-based LMs. For example, by integrating an 100-hidden layer RNNME model, the improvement in WRR is very close to a 500-hidden RNN model and even exceeds it by almost 2 points on *ALIF_test2*. The same observed when comparing BS-RNNME-100, BS-RNNME-300 and BS-RNN-700 results. This can be explained by the fact that the recurrent part of RNNME architecture focuses mainly on complementary information to the 5-order direct binary connections of the ME part. Although, this seems to be advantageous, the memory complexity of these models and so of n-grams is still much higher than that of RNN models.

6.4.3 Tuning decoding parameters

As presented in Section 6.3, the decoding algorithm uses the output probabilities of both OCR and LM components. We will study in this section the different hyper-parameters used for decoding, namely the LM weights ω_1 and ω_2, the rank-based and the score-based thresholds, th_{rk} and th_{sc}.

LM weights

To study the impact of LM weights, we have used the RNN-700 model. The emission probabilities of this model are incorporated in the decoding process with different values of ω_1 and ω_2. We set th_{rk} to 4 and we disable conditioning on hypothesis scores (th_{sc}

is not taken into account). We measure the WRR on the *DEV_Set* for each weight configuration. Results are presented by the color map in Figure. 6.6.

The map clearly shows the effect of the LM weights on the recognition results. Basically, very low values of ω_1 yield to low WRR. The analysis of the obtained results for $\omega_1 \in [0.6, 0.8]$ justifies the introduction of the second weight parameter ω_2. Best results are obtained with $\omega_1 = 0.7$. However, setting ω_2 to 0.55 instead of 0.7 allows an improvement of almost 0.3 points in terms of WRR (WRR passes from 87.5 to 87.8). For the rest of our experiments the couple of parameters (ω_1, ω_2) is set to $(0.7, 0.55)$.

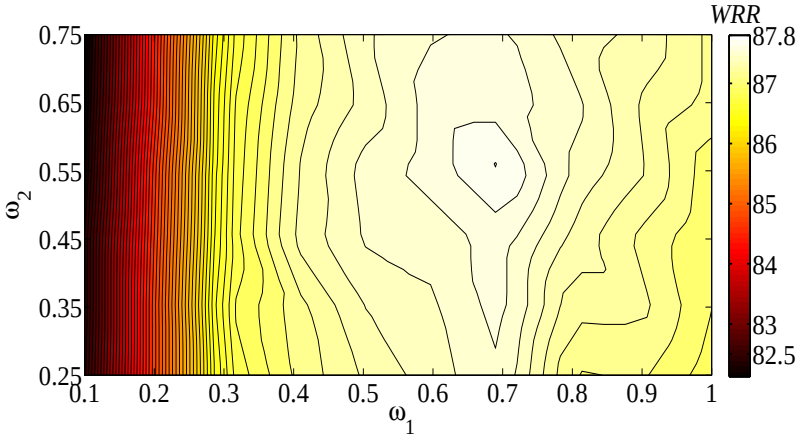
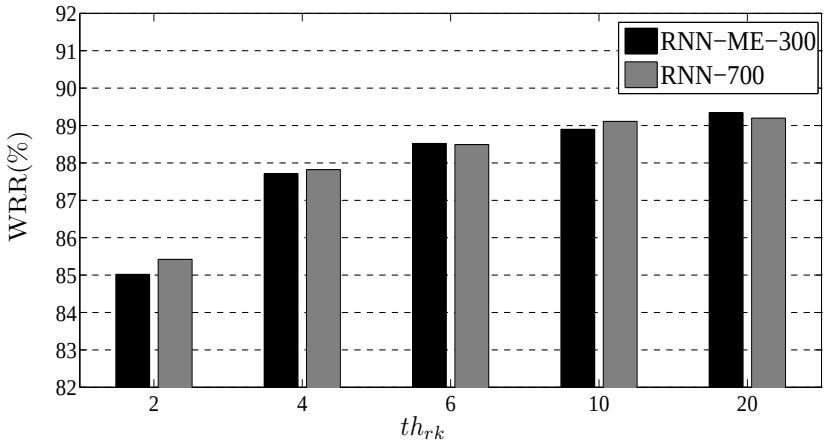


FIGURE 6.6: Recognition results variation with respect to LM weights.

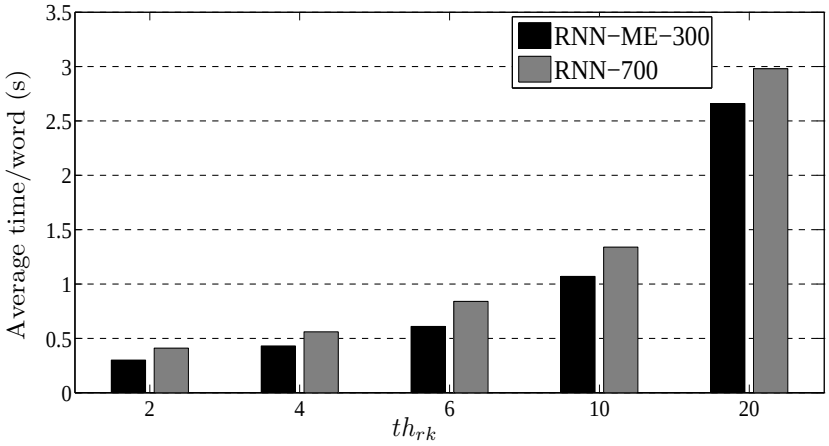
The beam width

To analyze the impact of the beam width on decoding results, we have considered the two best previously performing models: RNN-700 and RNNME-300. We set the th_{rk} to different values varying from 2 to 20 tolerated hypotheses by time-step. At this stage, no threshold on hypothesis scores is applied. We study the impact of the beam width both in terms of WRR and average processing time per word. This time is composed of the following: BLSTM stimulation of the input text features and the decoding algorithm that includes LM processing in order to determine LM-based conditional probabilities. Both results are presented in Figure 6.7.

Variations of WRR with respect to the beam size are presented in Figure 6.7(a). For both models, recognition results can be further improved by increasing the beam size. Passing from a beam of 2 to a beam of 20, the WRR increases by more than 4% for the RNNME-300 model and by almost 3.8% for RNN-700. However, a look into Figure 6.7(b) shows that this improvement costs in terms of processing time. For example, the obtained gain of 4% in WRR between $th_{rk} = 2$ and $th_{rk} = 20$ for the RNNME-300 model corresponds to a multiplication by 9 of the average processing time per word. A good tradeoff between WRR and the response time can be observed when th_{rk} varies between 2 and 6. For this interval, the average processing time is simply doubled for the two models but the WRR is improved by more than 3.5 points. However, beyond a beam width of 6, the improvement in terms of WRR becomes insignificant if we take into account the considerable increasing of the average processing time.



(a) WRR.



(b) Average processing time/word.

FIGURE 6.7: Impact of the beam width.

The score pruning

The th_{sc} parameter is introduced to retain only hypotheses with a probability score close to the best hypothesis for further consideration. In order to empirically study the effect of this criterion, we conduct several experiments on *DEV_SET* with different values of th_{sc} for each beam width. These values are fixed also in the logarithmic scale for better numerical representation.

First, we consider the BS-RNNME-300 schema. We measure both WRR and average response time per word for each (th_{rk}, th_{sc}) configuration. Results are illustrated in Table 6.2 and Table 6.3 respectively. Table 6.2 shows that decreasing the th_{sc} parameter reduces the recognition rates for different beam widths. In some cases, it can induce a little improvement by preventing some hypothesis, that are weak at a certain time step, to be extended to strong wrong hypotheses in the future due to high LM responses. However, in general, drastic reduction can be reached with very low score thresholds, namely for $th_{sc} = 2.5$. At the same time, the introduction of this threshold reduces the average processing time specially for high beam widths (cf. Table 6.3). For example, for $(th_{rk}, th_{sc}) = (20, 2.5)$, the average time per word has been divided by 8 while loosing

TABLE 6.2: Results - Impact of the score pruning in terms of WRR with the BS-RNNME-300 schema.

		th_{sc}				
		2.5	4.5	6.5	10.5	20.5
th_{rk}	2	83.7%	85.03%	**85.11%**	85.06%	85.03%
	4	84.72%	87.17%	**87.7%**	87.7%	87.7%
	6	84.86%	87.68%	**88.44%**	88.55%	88.52%
	10	84.86%	87.87%	**88.79%**	88.83%	**88.89%**
	20	84.86%	87.88%	**89.06%**	89.26%	**89.34%**

TABLE 6.3: Results - Impact of the score pruning in terms of average processing time per word with the BS-RNNME-300 schema.

		th_{sc}				
		2.5	4.5	6.5	10.5	20.5
th_{rk}	2	0.27s	0.26s	0.26s	0.28s	0.29s
	4	0.27s	0.29s	0.32s	0.36s	0.42s
	6	0.27s	0.29s	0.34s	0.56s	0.56s
	10	0.27s	0.31s	**0.37s**	0.57s	**0.93s**
	20	0.28s	0.34s	**0.46s**	0.9s	2.31s

almost 4 points in terms of WRR. However, the variation level of the WRR/time ratio is not the same for all the cases. We remark that above a score threshold of 6.5, the WRR remains almost stable for all beam widths while keeping a considerable reduction in response time. This becomes very remarkable specially for large beam widths. For example, with a beam of 20, the WRR still exceeds 89% while the average processing time per word is divided by 5. Therefore, with the score pruning, large beam widths (and thus best WRR) are no longer expensive in terms of response time.

Similarly, we conduct the same experiments on the *DEV_SET* with the BS-RNN-700 schema. Results are presented in Table 6.4 and Table 6.5. As illustrated in these tables, we can see that the previous observations remain valuable for this schema. The introduction of the th_{sc} allows keeping almost the same recognition results as those obtained with no score pruning (cf. Figure 6.7(a)) while reducing considerably the response time (5 times faster for $th_{sc} = 6.5$). Given these results, fixing the couple (th_{rk},th_{sc}) to (20,6.5) ensures the best tradeoff between recognition results and the average time per word.

6.4.4 Final results

The previous tuning experiments on the *DEV_SET* show better results with LM weights (ω_1,ω_2) and beam width and score thresholds (th_{rk},th_{sc}) equal to (0.7,0.55) and (20,6.5) respectively. In order to visualize the effective improvement compared to the primary

TABLE 6.4: Results - Impact of the score pruning in terms of WRR with the BS-RNN-700 schema.

		th_{sc}				
		2.5	4.5	6.5	10.5	20.5
th_{rk}	2	83.42%	85.5%	85.59%	85.42%	85.42%
	4	84.12%	87.08%	87.34%	87.42%	87.82%
	6	84.21%	87.7%	88.27%	88.49%	88.49%
	10	84.29%	88.01%	88.89%	89.03%	89.11%
	20	84.29%	88.1%	88.95%	89.2%	89.2%

TABLE 6.5: Results - Impact of the score pruning in terms of average processing time per word with the BS-RNN-700 schema.

		th_{sc}				
		2.5	4.5	6.5	10.5	20.5
th_{rk}	2	0.33s	0.32s	0.34s	0.39s	0.41s
	4	0.34s	0.42s	0.46s	0.54s	0.56s
	6	0.35s	0.43s	0.51s	0.69s	0.84s
	10	0.35s	0.43s	0.5s	0.79s	1.34s
	20	0.35s	0.43s	0.6s	1.19s	2.98s

results (cf. Section 6.4.2), we conduct further experiments with the new parameters on the test sets. Table 6.6 illustrates the obtained results.

The results show a considerable improvement in recognition rates over the basic recognition method that do not use linguistic information. This improvement reaches almost **16%** with the BS-RNN-700 schema on both datasets. The results show also that the character-level connectionist LMs still outperform the n-gram LM in terms of WRR. However, in terms of speed, the BS-7-gram is faster than other LMs given that probabilities are pre-stored in memory. Nevertheless, the BS-RNN-700 schema still requires much less in terms of memory. This model costs only a weights file of 4.5MB to be loaded without any pre-stotage of n-grams and their probabilities.

TABLE 6.6: Final results: Results are presented in terms of WRR / Average time per word and (ω_1, ω_2, th_{rk}, th_{sc}) are fixed to (0.7, 0.55, 20, 6.5).

Model	**Dataset**	
	ALIF-test1	***ALIF-test2***
BS-No-LM	73.53% / 0.2s	68.22% / 0.22s
BS-RNN-500	88.81% / 0.48s	84.03% / 0.58s
BS-RNN-700	89.31% / 0.55s	84.72% / 0.61s
BS-RNNME-300	88.63% / 0.46s	84.06% / 0.5s
BS-7-gram	85.5% / 0.44s	80.95% / 0.5s

As previously mentioned, the problem of Arabic text recognition in videos is much less addressed than Latin text recognition. Moreover, existing Arabic OCR systems are mainly dedicated to scanned document. Few works, if we do not say one or two works, have addressed 'modern' multimedia contents like natural images and videos. In addition, they reported results on different non public datasets. In this work, to state the performance of our proposed text recognizer, we performed a comparative study of it w.r.t. 'ABBYY Fine Reader 12' on the selected images of *ALIF_test1*. We remind that we have excluded examples with digits and punctuation marks. Results have been evaluated in terms of WRR as illustrated in Table 6.7. As we can see through this table, by introducing the RNN-based LM, we largely increase the gap between ABBYY results and ours to reach a difference of more than 35 points in terms of WRR.

TABLE 6.7: Comparative study on *ALIF_test1*.

Method	WRR
BS-No-LM	73.53%
BS-RNN-700	**89.31%**
BS-RNNME-300	**88.63%**
ABBYY Fine Reader 12	52.76%

Finally, we illustrate, in Figure 6.8, some of our OCR system outputs corrected by integrating the RNN-700 LM responses into the decoding schema. The linguistic information is able to correct confusions between similar characters like 'Laam' and 'Nuun' or 'Saad' and 'Ayn' in addition to the insertion and deletion cases produced by the BLSTM.

(a) 

(b) Kaaf Laam SPACE Alif Laam_Alif Laam Faa Alif Saad Alif Laam_Alif SPACE (كل الالفاصالا)

(c) Kaaf Laam SPACE Alif Laam_Alif Nuun Faa Ayn Alif Laam_Alif Taaa (كل الانفعالات)

(a) 

(b) Miim Alif Raa Alif Laam Jiim Ayn Alif Taaa (مارالجعات)

(c) Miim Raa Alif Jiim Ayn Alif Taaa (مراجعات)

(a) 

(b) Alif Haaa Baa Taaa SPACE Alif Laam Baa Haaa Raa (احبت البحر)

(c) Alif Haaa Baa Baa Taaa SPACE Alif Laam Baa Haaa Raa (احببت البحر)

FIGURE 6.8: Examples of text image recognition: (a) text image, (b) recognition with the BS-No-LM schema and (c) recognition with the BS-RNN-700 schema. Recognition results are given with both Latin and Arabic labellings.

6.5 Conclusion

In this chapter, we have introduced a methodology to improve BLSTM-CTC Arabic text recognition in videos using language modeling. We have focused on two main questions in order to reach our goal: (1) what type of language model to choose and (2) how to integrate it in the decoding schema?

First, we have proposed to take advantage from RNNs in order to model long range dependencies in the language. We have built two types of Arabic character-level language models: one is purely based on RNNs and the other is based on a joint learning of Maximum Entropy and RNN models. Second, we have introduced a decoding schema based on a modified version of the Beam Search algorithm. The proposed algorithm integrates linguistic information in parallel with the OCR responses. Although the OCR system is segmentation-free, the algorithm is able to join both the language model and the BLSTM responses at each time-step of the decoding without relying on a HMM-based system. We have introduced hyper-parameters to the decoding in order to reach better trade-off between recognition results and response time.

The whole paradigm has been extensively evaluated using our publicly available *ALIF dataset*. We have presented, besides, the experimental tuning of the used language models and decoding parameters. The chosen connectionist language models proved to outperform frequency-based n-grams by more than 4 points in terms of WRR. Furthermore, thanks to the contribution of these models and the impact of the decoding schema, we have reached an improvement of almost 16 points in WRR compared to the baseline BLSTM-CTC OCR system. The obtained results highlight the good recognition rates of our method that outperforms a well-known commercial OCR engine by almost 36%.

Chapter 7

Conclusion

In this thesis, we tackled the problem of Arabic embedded text detection and recognition in videos. Our goal was to provide efficient systems that can automatically extract and recognize Arabic textual tags overlaid in videos. With the actual huge growth of Arabic video contents, recognizing such text provides relevant information that can be very useful for video monitoring and structuring services.

We focused, in this work, on both Arabic text detection and recognition steps. The first step aims at searching and detecting Arabic text regions overlaid in videos and precisely localizing text lines. The second step consists in automatically transcribing the extracted text. Unlike text detection and reading in scanned document, dealing with video contents and Arabic texts raises additional challenges related mainly to the video environment (background complexity, varying colors, lighting, contrast, partial occlusion, etc.), the acquisition conditions (low resolution, blurring, noise, distortion, etc.) as well as to the text properties in general and Arabic text in particular (a wide variety of fonts, styles, sizes, scales, cursiveness of the Arabic script, frequent diacritics, morphologically rich characters, etc.). An additional major difficulty faced in this work is related to the lack of public Arabic text datasets issued from multimedia contents. This made difficult to build and evaluate Arabic video OCR systems. Practically, existing Arabic text datasets are limited to scanned documents for handwriting or printed text.

In this work, our first contributions are related to the detection step. We tackled the problem as language-specific issue. The proposed detection solutions are designed specifically for Arabic text avoiding, thus, a filtering step of non-Arabic scripts before the recognition stage. In order to face the wide complexity of the background and the Arabic text specificity, our methods are based on machine learning techniques. The methods do not rely on any pre-processing of the video frames nor on tedious morphological constraints. They rely, instead, on strong classification schemes based on sophisticated learning algorithms and well-designed training datasets. The problem is, first, seen as a text/non-text classification issue. The goal is to capture potential Arabic text regions in the video frame. We proposed three binary classifiers that learn discrimination between Arabic text regions and other patterns including non-text patterns and non-Arabic text patterns. The training set is made of text and non-text patches (positive and negative examples respectively). The two first classifiers are based on a multi-exit asymmetric boosting cascade where a strong classifier is hierarchically built through a pipeline of intermediate classifiers learned to reject non-text patterns. Text patches are presented as sets of hand-crafted features. For the first method, intermediate classifiers are based

on Gentleboost algorithm that learns to select relevant Muli-Block Local Binary Pattern features (MBLBP). This method was compared to the conventional boosting one widely used for pattern classification and which is based on Adaboost algorithm and Haar-like features. The third classification method relies on a Convolution Neural Network that learns both feature extraction, combination and classification in one single neural architecture. In order to raise the rejection ability of false alarms for the overall proposed methods, a bootstrapping procedure has been integrated into the training procedure. It keeps a dynamic negative training set with an ascending level of difficulty. In addition, for a better localization, the classifiers have been trained to reject hard non-text and non-Arabic text patterns by adding some examples into the negative training set like double-line patterns, badly cut text, Latin text, etc. As for the localization step, it is based on a multi-scale scanning scheme of the video frame followed by a clustering procedure of text candidates to form text lines. Evaluated on video frames issued from Arabic TV broadcast, the designed detectors showed high detection rates. However, higher accuracy and lower response time were achieved by the ConvNet-based method. This can be due to the ConvNet modeling of the problem that allows learning relevant features while learning the classification which leads to more discriminative features. Moreover, thanks to the spatial mapping technique used with the ConvNet technique for the localization, we avoided the tedious sliding window schema and reduced enormously the scanning time. As for the Boosting methods, they still very sensitive on what they receive as features which was depicted throw the obtained detection results over Haar-like and MBLBP features.

Regarding the recognition schema, we proposed to deal with the problem as a temporal classification issue. Given the cursiveness of the Arabic script and its morphological richness, a segmentation-based recognition approach may cause additional challenges specially with the complexity of the background. Instead, we proposed to transform extracted text images into sequences of features. Text recognition consists thus in sequentially labeling these features without prior segmentation. Instead of the cumbersome of the HMM infrastructure, this task is learned using a Bidirectional Long-Short Term Memory recurrent neural network (BLSTM) trained with a Connectionist Temporal Classification-based objective function (CTC). The network learns the temporal classification of the feature sequences using only unsegmented data. The text images are swept using multi-scale sliding windows where a feature generation model is applied at each position.

In this work, a special focus was made on the used features. Unlike existing methods that usually use hand-crafted features, we propose an auto-encoding schema at the character level that learns to infer relevant features without any labeled data nor priors. Independent of the recognition as a final step, the idea was to, separately, learn a representation of the Arabic text at elementary level (character for example) using an Auto-Encoder (AE) trained to reconstruct Arabic character images. The AE model is composed of an encoder that projects character images into a lower dimensional space (representation space) and a decoder that reconstruct these images using this projection vector seen as feature code. While training, the AE learns to extract relevant features for a better reconstruction that capture main specifications of Arabic characters. Given the wide variation of Arabic characters that depends on its position in a word, characters have been treated as glyphs. Two types of AEs have been built based on Deep Belief Networks (DBN) and simple Multi-Layer Perceptron (MLP). In addition, we proposed another feature generator model learned in an unsupervised manner. The model is based on Restricted Boltzman Machines (RBM) that takes as input character glyphs. Based

on an energy model, it learns a kind of confabulation between hidden and visible units. The model is not trained with a final target output but it explores a set of features in a layer-wise fashion. The hidden units of the final RBM have been considered as a feature vector. All these AEs and unsupervised models learn Arabic text features without any discrimination information between character glyphs. The whole temporal classification task (text recognition) is, thus, handled later in the sequence labeling stage by the BLSTM-CTC network. In order to further study the affect of the earlier discrimination between character glyphs at the feature generation stage, we proposed a third model based on a ConvNet that learns character classification using a set of labeled character images. While training, the model learns a set of relevant features that can be used to represent text images. The resulting recognition schemes, using each of the learned features, have been evaluated on text images issued from Arabic TV broadcast including very challenging conditions. High recognition rates have been obtained for all the methods. However, results showed that the BLSTM schema trained on ConvNet-based features achieved best results. In fact, these features have been learned with a classification goal which makes them more discriminative. The schema based on AEs presents very close results to ConvNet-based one. However, the AEs are models trained with a reconstruction goal. They learn to reproduce the overall information seen in the input including noise and distortions. Hence, the explored data representations may not reflect vary pointed features to be used by the BLSTM. As for the unsupervised model, the obtained results were very promising although they were the lowest. The model has been trained without any target output. Yet, the produced features have been successfully used by the BLSTM network and did not diverge during training. Compared to the ConvNet model, one major advantage of these AEs and the RBM models is that they do not rely on labeled data to be trained. Therefore, an upgrading of the training dataset may improve the results.

In addition to these recognition schemes, we proposed in this work other BLSTM-CTC based OCR systems that use hand-crafted and non-connectionist text features, namely raw features from binarized text images and Histogram of Oriented Gradient features. We conducted a comparative study between the overall recognition schemes and integrated an evaluation of a well-known commercial Arabic OCR engine. The obtained results revealed an outperformance of the proposed learned text features. The proposed deep connectionist models are more able to hierarchically represent the different modalities of the data. These deep architectures allow, moreover, to handle the non-linearity of the data patterns. In addition, for such models, all the parameters of the feature detectors are learned or in other terms 'dosed' according to what is seen in data. This makes them more robust to the wide complexity of the acquisition conditions and the Arabic text properties.

One other major contribution of this work is related to the used data. As we have mentioned, existing Arabic text datasets publicly available are limited to scanned documents. Therefore, in order to build and evaluate our text detection and recognition systems, we constructed our own datasets for both tasks. Using Arabic TV broadcast, we built different datasets manually annotated. The first dataset consists of video frames where Arabic text regions are precisely localized. This dataset have been used to build training Arabic text images for the proposed detection systems and also to evaluate them. The second dataset, called ALIF, has been dedicated for the recognition step. It consists of a large set of Arabic text images manually annotated that have been used to train and test our proposed recognition schemes. We made the ALIF dataset publicly and freely available for research purposes. To the best of our knowledge, this is the first publicly

available dataset in this context. It can provide a common evaluation and comparison ground for Arabic text recognition solutions. An additional dataset composed of labeled Arabic character images has been also built, in this work, to learn and test the feature extraction models. All the proposed detection and recognition datasets include a high variability in text properties and a considerable complexity in video environment and acquisition conditions. This explains a part of the robustness of our methods to these challenges.

In order to further improve recognition results, we proposed in addition to incorporate language modeling into the recognition schema. Our contribution in this part is related to the used language models (LM) and the integration schema of the linguistic information. We chose to take advantage from the recent advances in this field, namely LMs based on Recurrent Neural Networks (RNN) that are able to capture long range linguistic dependencies. We proposed different RNN architectures to learn Arabic character-level LMs and other architectures that combine RNNs with Maximum Entropy models. For the integration schema, we were not limited to a n-best rescoring of recognition results but instead we proposed a joint decoding based on a modified version of the Beam Search algorithm. The algorithm takes into account both the BLSTM and the LM output probabilities at each decoding time-step to extend a given number of the most likely hypotheses. We introduced, in addition, a set of hyper-parameters in order to improve recognition results and optimize the response time. An extensive experimental paradigm has been conducted in order to evaluate and compare the different LM architectures and to tune the decoding parameters. Results showed a considerable improvement in recognition results while keeping close space and time complexities of the decoding schema. The proposed connectionist LMs have been also compared to n-grams with the same decoding schema and have produced outstanding results.

Through this work, as we presented, we tackled almost the complete schema of Arabic embedded text reading in videos. Different methods were proposed for the detection, recognition and language modeling stages with and extensive comparative study between each set of methods. This allowed to draw conclusions about the proposed solutions while reaching outstanding results at each stage. A special focus has been also given to the datasets which allowed us to build and publish new multimedia Arabic text dataset that may serve the OCR community in future works. Thanks to the well-designed data that present large range of challenges (hard video environment, acquisition conditions and complex text properties) and also to the designed detection and recognition models, we reached outstanding results. Evaluation on test sets issued from Arabic TV broadcast showed a text detection rate of almost 97% with only few false alarms. As for the recognition solutions, we reached a maximum word recognition rate (WRR) over 71% without language modeling. The integration of the linguistic information has improved this rate by more than 16% to reach a final WRR of almost 89% without relying on any pre-processing of the input video frames.

Perspectives

This work can be further improved by investigating some critical points at each stage of the OCR schema. First for the detection step, results can be boosted if we take into account the temporal correlation between successive video frames. Using for example the Multiple Frame Integration technique (MFI) [HYZ02], the separation between text and background pixels can be further enhanced, specially for a moving background. Moreover, introducing a feedback procedure from the recognition step can improve the precision of the text localization specially for borders of the detected text regions. This

can also reduce a considerable number of false alarms caused by a split case or a non-Arabic text detection, etc.

Second, for the recognition step, we used very powerful deep networks that used to produce good results in many computer vision tasks. These models are able to capture very complex data modalities and generate relevant features. Similarly, for the BLSTM network that can learn long-range data dependencies. On the one hand, we avoided, in this work, all the hand-designed operations that usually require a binarization step which is not a trivial task in our context. On the other hand, a large part of challenges have been faced thanks to the connectionist deep models and the training procedure. However, these models, including ConvNets, DBN, RBM, MLPs and RNNs, are still very dependent on what they receive as data. Therefore, increasing the amount of our training sets may enhance recognition results. First, this makes the dataset much more exhaustive by equally focusing on every challenge (varying text properties and different acquisition conditions) for better robustness of the proposed models. Second, using larger datasets allows to employ more complex and larger network architectures which enhance the models generalization and accuracy. In particular for the BLSTM network, focusing on this couple data/architecture, can avoid many over-fitting cases during the training. The features that the BLSTM receives as input may provide very relevant 'local' presentation of the handled text image. However, if the training text images do not sufficiently cover the variations of sequential data modalities and mainly larger set of 'combinations' between characters, the network can not get to learn many labeling configurations from the sequence of features and falls in an over-fitting. Increasing training data may be also very interesting for the unsupervised text feature generation using the RBM schema. Since we do not have to use labeled text regions, we can train the model to extract features from sub-character, paws and even word images. However, the BLSTM still requires larger training set. We can also train this latter on recognizing paws in addition to character at each time-step which may avoid missing very thin characters and may help to infer higher level data interdependencies.

One other possible alternative for text recognition is to avoid the prior text feature extraction step. This can be achieved by training a Multi-Dimensional BLSTM network [GS09] directly on raw text images. This can be very efficient if we opt for a binarization step handling thus only black and white pixels. However, in our case with very complex background, it will be more attractive and interesting if we avoid the binarization. This surely requires more annotated text images for training and a special focus on the used BLSTM architecture and connections to handle the spatio-temporal dimensions present in the data.

Recognition and also detection results can be further enhanced by applying some super-resolution techniques. These techniques aim at increasing the resolution of text images in order to provide images with better quality. This can handle a large part of challenges related to the video acquisition conditions. It may enhance recognition by providing clearer text regions specially for Arabic character diacritics and also for critical character strokes.

Regarding language modeling, we proposed in this work to handle this component as a part of the recognition and not as a post-processing step. Results have been considerably improved using only character-level LMs. They can be further enhanced by introducing a word-level LM and dictionaries information into the decoding schema. However, unlike Latin languages, building a proper Arabic word-level LM may require a pre-processing

step of the Arabic text using large set of syntax and grammar rules for a better representation of the target vocabulary. In addition, another idea to explore is to optimize the whole parameters (of the decoding and the language models) at once with an novel and appropriate objective function.

Bibliography

[AAKM+12] I. Aljarrah, O. Al-Khaleel, K. Mhaidat, M. Alrefai, A. Alzu'bi, M. Rabab'ah, et al. Automated system for arabic optical character recognition with lookup dictionary. *Journal of Emerging Technologies in Web Intelligence*, 4(4):362–370, 2012.

[AC99] S. Antani and D. Crandall. Extraction of text in video. In *International Conference on Pattern Recognition*, 1999.

[AGP07] M. Anthimopoulos, B. Gatos, and I. Pratikakis. Multiresolution text detection in video frames. In *VISAPP (2)*, pages 161–166, 2007.

[AGP10] M. Anthimopoulos, B. Gatos, and I. Pratikakis. A two-stage scheme for text detection in video images. *Image and Vision Computing*, 28(9):1413–1426, 2010.

[Alg13] Y.M. Alginahi. A survey on arabic character segmentation. *International Journal on Document Analysis and Recognition (IJDAR)*, 16(2):105–126, 2013.

[AMMQ08] H.A. Al-Muhtaseb, S.A. Mahmoud, and R.S. Qahwaji. Recognition of off-line printed arabic text using hidden markov models. *Signal Processing*, 88(12):2902–2912, 2008.

[AOCS00] Y. Al-Ohali, M. Cheriet, and C. Suen. Databases for recognition of handwritten arabic cheques. In *International Workshop on Frontiers in Handwriting Recognition*, pages 601–606, 2000.

[ARFM13] I. Ahmad, L. Rothacker, G.A. Fink, and S.A. Mahmoud. Novel sub-character hmm models for arabic text recognition. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 658–662, 2013.

[ASB11] M. Abbas, K. Smaïli, and D. Berkani. Evaluation of topic identification methods on arabic corpora. *JDIM*, 9(5):185–192, 2011.

[BCNN13] A. Bissacco, M. Cummins, Y. Netzer, and H. Neven. Photoocr: Reading text in uncontrolled conditions. In *International Conference on Computer Vision*, pages 785–792, 2013.

[BDVJ03] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[Ben99] Y. Bengio. Markovian models for sequential data. *Neural computing surveys*, 2(1049):129–162, 1999.

[Ber86] J. Bernsen. Dynamic thresholding of grey-level images. In *International conference on pattern recognition*, pages 1251–1255, 1986.

[Bis95] C.M. Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.

[Bis01] C.M. Bishop. Bishop pattern recognition and machine learning, 2001.

[BLP+07] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, et al. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.

[BMB13] S. Banerjee, K. Mullick, and U. Bhattacharya. A robust approach to extraction of texts from camera captured images. In *International Workshop on Camera-Based Document Analysis and Recognition*, pages 30–46. 2013.

[Bod02] M. Boden. A guide to recurrent neural networks and backpropagation. *The Dallas project, SICS technical report*, 2002.

[BPP96] A.L. Berger, V.J.D. Pietra, and S.A.D. Pietra. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):39–71, 1996.

[Bre08] T.M. Breuel. The ocropus open source ocr system. In *Electronic Imaging 2008*, pages 68150F–68150F. International Society for Optics and Photonics, 2008.

[BS97] B. Bushofa and M. Spann. Segmentation and recognition of arabic characters by structural classification. *Image and Vision Computing*, 15(3):167–179, 1997.

[BSF94] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[BUHAAAS13] T.M. Breuel, A. Ul-Hasan, M. Ali Al-Azawi, and F. Shafait. High-performance ocr for printed english and fraktur using lstm networks. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 683–687, 2013.

[CAK03] D. Crandall, S. Antani, and R. Kasturi. Extraction of special effects caption text events from digital video. *International Journal on Document Analysis and Recognition*, 5(2-3):138–157, 2003.

[CCC+11] A. Coates, B. Carpenter, C. Case, S. Satheesh, B. Suresh, T. Wang, D.J. Wu, and A.Y. Ng. Text detection and character recognition in scene images with unsupervised feature learning. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 440–445, 2011.

[CG99] S.F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393, 1999.

[Che03] D. Chen. Text detection and recognition in images and video sequences. Technical report, 2003.

[Che09] S.F. Chen. Shrinking exponential language models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 468–476, 2009.

[Cho13] K. Cho. Boltzmann machines and denoising autoencoders for image denoising. *arXiv preprint arXiv:1301.3468*, 2013.

[CMMS12] D. Cireşan, U. Meier, J. Masci, and J. Schmidhuber. Multi-column deep neural network for traffic sign classification. *Neural Networks*, 32:333–338, 2012.

[CMR+09] S.F. Chen, L. Mangu, B. Ramabhadran, R. Sarikaya, and A. Sethy. Scaling shrinkage-based language models. In *IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 299–304, 2009.

[CSL02] M. Cai, J. Song, and M.R. Lyu. A new approach for video text detection. In *International Conference on Image Processing*, volume 1, pages 117–120, 2002.

[CWW15] C. Chen, D-H. Wang, and H. Wang. Scene character and text recognition: The state-of-the-art. In *Image and Graphics*, pages 310–320. 2015.

[DC93] D. Demers and G. Cottrell. Non-linear dimensionality reduction. In *Advances in Neural Information Processing Systems 5*, pages 580–587, 1993.

[DG08] M. Delakis and C. Garcia. text detection with convolutional neural networks. In *VISAPP (2)*, pages 290–294, 2008.

[DH97] R. Davidson and R. Hopely. Arabic and persian ocr training and test data sets. In *Proc. Symp. On Document Image Understanding Technology*, 1997.

[DT05] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893, 2005.

[EBCBGMZM11] S. Espana-Boquera, M.J. Castro-Bleda, J. Gorbe-Moya, and F. Zamora-Martinez. Improving offline handwritten text recognition with hybrid hmm/ann models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(4):767–779, 2011.

[EGMS14] K. Elagouni, C. Garcia, F. Mamalet, and P. Sébillot. Text recognition in multimedia documents: a study of two neural-based ocrs using and avoiding character segmentation. *International Journal on Document Analysis and Recognition (IJDAR)*, 17(1):19–31, 2014.

[EGS11] K. Elagouni, C. Garcia, and P. Sébillot. A comprehensive neural-based approach for text recognition in videos using natural language processing. In *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, page 23, 2011.

[Elk03] C. Elkan. Using the triangle inequality to accelerate k-means. In *ICML*, volume 3, pages 147–153, 2003.

[Elm90] J.L. Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[EOW10] B. Epshtein, E. Ofek, and Y. Wexler. Detecting text in natural scenes with stroke width transform. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2963–2970, 2010.

[ES02] D. Eck and J. Schmidhuber. Finding temporal structure in music: Blues improvisation with lstm recurrent networks. In *IEEE Workshop on Neural Networks for Signal Processing*, pages 747–756, 2002.

[FE73] M.A. Fischler and R.A. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on computers*, (1):67–92, 1973.

[FGS07] S. Fernández, A. Graves, and J. Schmidhuber. An application of recurrent neural networks to discriminative keyword spotting. In *Artificial Neural Networks–ICANN 2007*, pages 220–229. 2007.

[FHT+00] J. Friedman, T. Hastie, R. Tibshirani, et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407, 2000.

[FM82] K. Fukushima and S. Miyake. Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern recognition*, 15(6):455–469, 1982.

[FSA99] Y. Freund, R. Schapire, and N. Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[Fuk75] K. Fukushima. Cognitron: A self-organizing multilayered neural network. *Biological cybernetics*, 20(3-4):121–136, 1975.

[Fuk88] K. Fukushima. A neural network for visual pattern recognition. *Computer*, 21(3):65–75, 1988.

[GD04] C. Garcia and M. Delakis. Convolutional face finder: A neural architecture for fast and robust face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1408–1423, 2004.

[GFGS06] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376, 2006.

[GFS05] A. Graves, S. Fernández, and J. Schmidhuber. Bidirectional lstm networks for improved phoneme classification and recognition. In *Artificial Neural Networks: Formal Models and Their Applications–ICANN 2005*, pages 799–804. 2005.

[GJ14] A. Graves and N. Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1764–1772, 2014.

[GJM13] A. Graves, N. Jaitly, and A.-A. Mohamed. Hybrid speech recognition with deep bidirectional lstm. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 273–278, 2013.

[GLB+08] A. Graves, M. Liwicki, H. Bunke, J. Schmidhuber, and S. Fernández. Unconstrained on-line handwriting recognition with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 577–584, 2008.

[GLF+09] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):855–868, 2009.

[Gra12a] A. Graves. Offline arabic handwriting recognition with multidimensional recurrent neural networks. In *Guide to OCR for Arabic Scripts*, pages 297–313. 2012.

[Gra12b] A. Graves. *Supervised sequence labelling*. Springer, 2012.

[Gra13] A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[GS05] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610, 2005.

[GS09] A. Graves and J. Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in neural information processing systems*, pages 545–552, 2009.

[GSC00] F.A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

[GSS03] F.A. Gers, N.N. Schraudolph, and J. Schmidhuber. Learning precise timing with lstm recurrent networks. *The Journal of Machine Learning Research*, 3:115–143, 2003.

[Hin02] G.E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[Hin07] G. E. Hinton. To recognize shapes first learn to generate images. In *Computational Neuroscience: Theoretical Insights into Brain Function*, volume 165, pages 535–547. 2007.

[HMZ09] X. Huang, H. Ma, and H. Zhang. A new video text extraction approach. In *International Conference on Multimedia and Expo*, pages 650–653, 2009.

[Hop82] J.J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[HQT14] W. Huang, Y. Qiao, and X. Tang. Robust scene text detection with convolution neural network induced mser trees. In *Computer Vision–ECCV 2014*, pages 497–511. 2014.

[HS97] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[HS06] G.E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[HSU13] R. Huang, P. Shivakumara, and S. Uchida. Scene character detection by an edge-ray filter. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 462–466, 2013.

[HW62] D.H. Hubel and T.N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154, 1962.

[HYZ02] X-S. Hua, P. Yin, and H-J. Zhang. Efficient video text recognition using multiple frame integration. In *International Conference on Image Processing*, volume 2, pages 397–400, 2002.

[JKJ04] K. Jung, K.I. Kim, and A.K. Jain. Text information extraction in images and video: a survey. *Pattern recognition*, 37(5):977–997, 2004.

[JKK+13] F. K. Jaiem, S. Kanoun, M. Khemakhem, H. Abed, and J. Kardoun. Database for arabic printed text recognition research. In *International Conference on Image Analysis and Processing*, pages 251–259, 2013.

[JMRS] F. Jelinek, B. Merialdo, S. Roukos, and M. Strauss. A dynamic language model for speech recognition. In *HLT*, volume 91, pages 293–295.

[JSVZ16] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Reading text in the wild with convolutional neural networks. *International Journal of Computer Vision*, 116(1):1–20, 2016.

[Kat87] S.M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(3):400–401, 1987.

[Kho02] M.S. Khorsheed. Off-line arabic character recognition–a review. *Pattern analysis & applications*, 5(1):31–45, 2002.

[KK09] W. Kim and C. Kim. A new approach for overlay text detection and extraction from complex video scene. *IEEE Transactions on Image Processing*, 18(2):401–411, 2009.

[KMKB11] S. Kombrink, T. Mikolov, M. Karafiát, and L. Burget. Recurrent neural network based language modeling in meeting recognition. In *INTERSPEECH*, pages 2877–2880, 2011.

[KN95] R. Kneser and H. Ney. Improved backing-off for m-gram language modeling. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 181–184, 1995.

[Koh90] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[L+89] Y. LeCun et al. Generalization and network design strategies. *Connections in Perspective*, pages 143–55, 1989.

[LB95] Y. LeCun and Y. Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):255–258, 1995.

[LBBH98] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LBH15] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[LCL00] Y.-K. Lim, S.-H. Choi, and S.-W. Lee. Text extraction in mpeg compressed video for content-based indexing. In *International Conference on Pattern Recognition*, volume 4, pages 409–412, 2000.

[LDH+90] B.B. LeCun, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, 1990.

[LDK00] H. Li, D. Doermann, and O. Kia. Automatic text detection and tracking in digital video. *IEEE Transactions on Image Processing*, 9(1):147–156, 2000.

[LDL05] J. Liang, D. Doermann, and H. Li. Camera-based analysis of text and documents: a survey. *International Journal of Document Analysis and Recognition (IJDAR)*, 7(2-3):84–104, 2005.

[LeC86] Y. LeCun. Learning process in an asymmetric threshold network. In *Disordered systems and biological organization*, pages 233–240. 1986.

[LG06] L. M Lorigo and V. Govindaraju. Offline arabic handwriting recognition: a survey. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (5):712–724, 2006.

[LGTB97] S. Lawrence, C.L. Giles, A.C. Tsoi, and A.D. Back. Face recognition: A convolutional neural-network approach. *IEEE Transactions on Neural Networks*, 8(1):98–113, 1997.

[Lie03] R. Lienhart. Video ocr: A survey and practitioner's guide. In *Video mining*, pages 155–183. 2003.

[Liu08] X. Liu. A camera phone based currency reader for the visually impaired. In *Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility*, pages 305–306, 2008.

[LLL+11] J.-J. Lee, P.-H. Lee, S.-W. Lee, A. Yuille, and C. Koch. Adaboost for text detection in natural scene. In *International Conference on Document Analysis and Recognition*, pages 429–434, 2011.

[LLP96] S.-W. Lee, D.-J. Lee, and H.-S. Park. A new methodology for gray-scale character segmentation and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10):1045–1050, 1996.

[LMA15] H. Luqman, S.A. Mahmoud, and S. Awaida. Arabic and farsi font recognition: Survey. *International Journal of Pattern Recognition and Artificial Intelligence*, 29(01), 2015.

[LS12] D.-S. Lee and R. Smith. Improving book ocr by adaptive language and image models. In *Document Analysis Systems (DAS), 2012 10th IAPR International Workshop on*, pages 115–119, 2012.

[LSC05] M.R. Lyu, J. Song, and M. Cai. A comprehensive method for multilingual video text detection, localization, and extraction. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(2):243–255, 2005.

[LW02] R. Lienhart and A. Wernicke. Localizing and segmenting text in images and videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(4):256–268, 2002.

[MAJ12a] A. Mishra, K. Alahari, and C.V. Jawahar. Scene text recognition using higher order language priors. In *BMVC 2012-23rd British Machine Vision Conference*. BMVA, 2012.

[MAJ12b] A. Mishra, K. Alahari, and C.V. Jawahar. Top-down and bottom-up cues for scene text recognition. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2687–2694, 2012.

[MBHV12] A. M. Alimi M. Ben Halima, H. Karray and A. Fernández Vila. Nf-savo: Neuro-fuzzy system for arabic video ocr. *International Journal of Advanced Computer Science and Applications*, 3(10):128–136, 2012.

[MBLD91] O. Matan, C.J.C. Burges, Y. LeCun, and J.S. Denker. Multi-digit recognition using a space displacement neural network. In *NIPS*, pages 488–495, 1991.

[MBN+13] S. Milyaev, O. Barinova, T. Novikova, P. Kohli, and V. Lempitsky. Image binarization for end-to-end text understanding in natural images. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 128–132, 2013.

[MCA+10] R. F. Moghaddam, M. Cheriet, M. M. Adankon, K. Filonenko, and R. Wisnovsky. Ibn sina: A database for research on processing and understanding of arabic manuscripts images. In *IAPR International Workshop on Document Analysis Systems*, pages 11–18, 2010.

[MDK+11] T. Mikolov, A. Deoras, S. Kombrink, L. Burget, and J. Cernockỳ. Empirical evaluation and combination of advanced language modeling techniques. In *INTERSPEECH*, number s 1, pages 605–608, 2011.

[MEA12] V. Märgner and H. El Abed. *Guide to OCR for Arabic scripts*. Springer, 2012.

[Mik12] T. Mikolov. *Statistical Language Models Based on Neural Networks*. PhD thesis, Ph. D. thesis, Brno University of Technology, 2012.

[MKB+10] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, volume 2, page 3, 2010.

[MKKA12] A. Mezghani, S. Kanoun, M. Khemakhem, and H. Abed. A database for arabic handwritten text image recognition and writer identification. In *International Conference on Frontiers in Handwriting Recognition*, pages 399–402, 2012.

[MMO10] M. Moradi, S. Mozaffari, and A.A. Orouji. Farsi/arabic text extraction from video images by corner detection. In *Iranian Conference on Machine Vision and Image Processing*, pages 1–6, 2010.

[MNY99] S. Mori, H. Nishida, and H. Yamada. *Optical character recognition*. John Wiley & Sons, Inc., 1999.

[MPR02] M. Mohri, F. Pereira, and M. Riley. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88, 2002.

[MPR05] R. Mehran, H. Pirsiavash, and F. Razzazi. A front-end ocr for omnifont persian/arabic cursive printed documents. In *Digital Image Computing: Techniques and Applications*, pages 56–56, 2005.

[MZ12] T. Mikolov and G. Zweig. Context dependent recurrent neural network language model. In *SLT*, pages 234–239, 2012.

[MZJ+07] G. Miao, G. Zhu, S. Jiang, Q. Huang, C. Xu, and W. Gao. A real-time score detection and recognition approach for broadcast basketball video. In *International Conference on Multimedia and Expo*, pages 1691–1694, 2007.

[NBKL12] T. Novikova, O. Barinova, P. Kohli, and V. Lempitsky. Large-lexicon attribute-consistent text recognition in natural images. In *Computer Vision–ECCV 2012*, pages 752–765. 2012.

[NGP11] K. Ntirogiannis, B. Gatos, and I. Pratikakis. Binarization of textual content in video frames. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 673–677, 2011.

[Nib85] W. Niblack. *An introduction to digital image processing*. Strandberg Publishing Company, 1985.

[NM10] L. Neumann and J. Matas. A method for text localization and recognition in real-world images. In *Computer Vision–ACCV 2010*, pages 770–783. 2010.

[NM13a] L. Neumann and J. Matas. On combining multiple segmentations in scene text recognition. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 523–527, 2013.

[NM13b] L. Neumann and J. Matas. Scene text localization and recognition with oriented stroke detection. In *International Conference on Computer Vision*, pages 97–104, 2013.

[NS08] D. Nistér and H. Stewénius. Linear time maximally stable extremal regions. In *Computer Vision–ECCV 2008*, pages 183–196. 2008.

[NUS+15] S. Naz, A.I. Umar, S.H. Shirazi, S.B. Ahmed, M.I. Razzak, and I. Siddiqi. Segmentation techniques for recognition of arabic-like scripts: A comprehensive survey. *Education and Information Technologies*, pages 1–17, 2015.

[OLM07] M. Osadchy, Y. LeCun, and M.L. Miller. Synergistic face detection and pose estimation with energy-based models. *The Journal of Machine Learning Research*, 8:1197–1215, 2007.

[OPH94] T. Ojala, M. Pietikainen, and D. Harwood. Performance evaluation of texture measures with classification based on kullback discrimination of distributions. In *Pattern Recognition, 1994. Vol. 1-Conference A: Proceedings of the 12th IAPR International Conference on Computer Vision & Image Processing*, volume 1, pages 582–585, 1994.

[OPH96] T. Ojala, M. Pietikäinen, and D. Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern recognition*, 29(1):51–59, 1996.

[Ots75] N. Otsu. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975.

[PBQT12] J. Poignant, L. Besacier, G. Quénot, and F. Thollard. From text detection in videos to person identification. In *International Conference on Multimedia and Expo (ICME)*, pages 854–859, 2012.

[PCPN11] X. Peng, H. Cao, R. Prasad, and P. Natarajan. Text extraction from video using conditional random fields. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1029–1033, 2011.

[PCS+13] X. Peng, H. Cao, S. Setlur, V. Govindaraju, and P. Natarajan. Multilingual ocr research and applications: an overview. In *Proceedings of the 4th International Workshop on Multilingual OCR*, page 1, 2013.

[PDSR11] C. Parada, M. Dredze, A. Sethy, and A. Rastrow. Learning subword units for open vocabulary speech recognition. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 712–721. Association for Computational Linguistics, 2011.

[PHC08] M.-T. Pham, V.-D.D. Hoang, and T.J. Cham. Detection with multi-exit asymmetric boosting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[PLDW06] L. Peng, C. Liu, X. Ding, and H. Wang. Multilingual document recognition research and its application in china. In *International Conference on Document Image Analysis for Libraries*, pages 126–132, 2006.

[PLK+10] J. Park, G. Lee, E. Kim, J. Lim, S. Kim, H. Yang, M. Lee, and S. Hwang. Automatic detection and recognition of korean text in outdoor signboard images. *Pattern Recognition Letters*, 31(12):1728–1739, 2010.

[PM13] M. Tanvir Parvez and S.A. Mahmoud. Offline arabic handwritten text recognition: a survey. *ACM Computing Surveys (CSUR)*, 45(2):23, 2013.

[PMM+02] M. Pechwitz, S. Snoussi Maddouri, V. Margner, N. Ellouze, and H. Amiri. Ifn/enit database of handwritten arabic words. In *Colloque International Francophone sur l'Écrit et le Document (CIFED)*, pages 129–136, 2002.

[PSST11] T.Q. Phan, P. Shivakumara, B. Su, and C.L. Tan. A gradient vector flow-based method for video character segmentation. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1024–1028, 2011.

[PST09] T.Q. Phan, P. Shivakumara, and C.L. Tan. A laplacian method for video text detection. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 66–70, 2009.

[Rab89] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[RHW85] D. E Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. Technical report, DTIC Document, 1985.

[RHW86] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[RN95] S. Russell and P. Norvig. A modern approach. *Artificial Intelligence*, 25:27, 1995.

[Rob94] A.J. Robinson. An application of recurrent nets to phone probability estimation. *IEEE Transactions on Neural Networks*, 5(2):298–305, 1994.

[Ros94] R. Rosenfeld. *Adaptive Statistical Language Modeling: A Maximum Entropy Approach. Pittsburgh.* PhD thesis, 1994.

[RPML95] K. Romeo-Pakker, H. Miled, and Y. Lecourtier. A new approach for latin/arabic character segmentation. In *International Conference on Document Analysis and Recognition*, volume 2, pages 874–877, 1995.

[RRS+13] S. Roy, P.P. Roy, P. Shivakumara, G. Louloudis, C.L. Tan, and U. Pal. Hmm-based multi oriented text recognition in natural scene image. In *Asian Conference on Pattern Recognition (ACPR)*, pages 288–292, 2013.

[RSGP15] J. A. Rodriguez-Serrano, A. Gordo, and F. Perronnin. Label embedding: A frugal baseline for text recognition. *International Journal of Computer Vision*, 113(3):193–207, 2015.

[RSRvdN13] S.F. Rashid, M.-P. Schambach, J. Rottland, and S. von der Nüll. Low resolution arabic recognition with multidimensional recurrent neural networks. In *International Workshop on Multilingual OCR*, page 6:1–6:5, 2013.

[SA10] M. Saad and W. Ashour. Arabic morphological tools for text mining. In *International Symposium on Electrical and Electronics Engineering and Computer Science*, pages 112–117, 2010.

[Sch03] R.E. Schapire. The boosting approach to machine learning: An overview. In *Nonlinear estimation and classification*, pages 149–171. Springer, 2003.

[SCS09] T. Som, D. Can, and M. Saraçlar. Hmm-based sliding video text recognition for turkish broadcast news. In *International Symposium on Computer and Information Sciences*, pages 475–479, 2009.

[SG02] H. Schwenk and J.-L. Gauvain. Connectionist language modeling for large vocabulary continuous speech recognition. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages I–765, 2002.

[SG07a] Z. Saidane and C. Garcia. Automatic scene text recognition using a convolutional neural network. In *International Workshop on Camera-Based Document Analysis and Recognition*, volume 1, 2007.

[SG07b] Z. Saidane and C. Garcia. Robust binarization for video text recognition. In *International Conference on Document Analysis and Recognition (ICDAR)*, volume 2, pages 874–879, 2007.

[SG08] Z. Saidane and C. Garcia. An automatic method for video character segmentation. In *Image Analysis and Recognition*, pages 557–566. Springer, 2008.

[SGD09] Z. Saidane, C. Garcia, and J-L. Dugelay. The image text recognition graph (itrg). In *International Conference on Multimedia and Expo*, pages 266–269, 2009.

[SH94] A.W. Senior and T. Hall. *Offline cursive handwriting recognition using recurrent neural networks*. PhD thesis, PhD thesis, Cambridge University Engineering, 1994.

[Sha08] Z. Shaaban. A new recognition scheme for machine-printed arabic texts based on neural networks. In *Proceedings of World Academy of Science, Engineering and Technology*, volume 31, pages 25–27, 2008.

[SHT08] P. Shivakumara, W. Huang, and C.L. Tan. Efficient video text detection using edge features. In *International Conference on Pattern Recognition*, pages 1–4, 2008.

[Sin13] S. Singh. Optical character recognition techniques: a survey. *Journal of emerging Trends in Computing and information Sciences*, 4(6):545–550, 2013.

[SKA+13] F. Slimane, S. Kanoun, H. El Abed, A. M. Alimi, R. Ingold, and J. Hennebert. Icdar2013 competition on multi-font and multi-size digitally represented arabic text. In *International Conference on Document Analysis and Recognition*, pages 1433–1437, 2013.

[SKCL13] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun. Pedestrian detection with unsupervised multi-stage feature learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3626–3633, 2013.

[SKEA+11] F. Slimane, S. Kanoun, H. El Abed, A.M. Alimi, R. Ingold, and J. Hennebert. Icdar 2011-arabic recognition competition: Multi-font multi-size digitally represented text. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1449–1453, 2011.

[SKH+13] F. Slimane, S. Kanoun, J. Hennebert, A.M. Alimi, and R. Ingold. A study on font-family and font-size recognition applied to arabic word images at ultra-low resolution. *Pattern Recognition Letters*, 34(2):209–218, 2013.

[SKHS98] T. Sato, T. Kanade, E.K. Hughes, and M.A. Smith. Video ocr for digital news archive. In *International Workshop on Content-Based Access of Image and Video Database*, pages 52–60, 1998.

[SKPK00] C. Shin, K. Kim, M. Park, and H.J. Kim. Support vector machine-based text detection in digital video. In *IEEE Signal Processing Society Workshop on Neural networks for signal processing*, volume 2, pages 634–641, 2000.

[SL14] B. Su and S. Lu. Accurate scene text recognition based on recurrent neural network. In *Computer Vision–ACCV 2014*, pages 35–48. 2014.

[SP97] M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[SP00] J. Sauvola and M. Pietikäinen. Adaptive document image binarization. *Pattern recognition*, 33(2):225–236, 2000.

[SPB12] N. Sharma, U. Pal, and M. Blumenstein. Recent advances in video based document processing: a review. In *International Workshop on Document Analysis Systems (DAS)*, pages 63–68, 2012.

[SR96] A. Senior and T. Robinson. Forward-backward retraining of recurrent neural networks. *Advances in Neural Information Processing Systems*, pages 743–749, 1996.

[SS99] R.E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37(3):297–336, 1999.

[SSN12] M. Sundermeyer, R. Schlüter, and H. Ney. Lstm neural networks for language modeling. In *INTERSPEECH*, pages 194–197, 2012.

[SSP03] P.Y. Simard, D. Steinkraus, and J.C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *International Conference on Document Analysis and Recognition*, pages 958–963, 2003.

[SSP+12] P. Shivakumara, R.P. Sreedhar, T.Q. Phan, S. Lu, and C.L. Tan. Multioriented video scene text detection through bayesian classification and boundary growing. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(8):1227–1235, 2012.

[Sto02] A. Stolcke. Srilm-an extensible language modeling toolkit. In *INTERNATIONAL CONFERENCE ON SPOKEN LANGUAGE PROCESSING*, pages 901–904, 2002.

[SWX+13] C. Shi, C. Wang, B. Xiao, Y. Zhang, S. Gao, and Z. Zhang. Scene text recognition using part-based tree-structured character detection. In *International Conference on Computer Vision and Pattern Recognition*, pages 2961–2968, 2013.

[SZGH09] T.-H. Su, T.-W. Zhang, D.-J. Guan, and H.-J. Huang. Off-line recognition of realistic chinese handwriting using segmentation-free strategy. *Pattern Recognition*, 42(1):167–182, 2009.

[Szö10] I. Szöke. *Hybrid word-subword spoken term detection*. PhD thesis, Brno University of Technology, 2010.

[TF96] B. Timsari and H. Fahimi. Morphological approach to character recognition in machine-printed persian words. In *Electronic Imaging: Science & Technology*, pages 184–191, 1996.

[TGJ+15] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. Efficient object localization using convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 648–656, 2015.

[THR+13] N. Tomeh, N. Habash, R. Roth, N. Farra, P. Dasigi, and M. Diab. Reranking with linguistic and semantic features for arabic optical character recognition. In *ACL (2)*, pages 549–555, 2013.

[TSAA93] M. Tellache, M.A. Sid-Ahmed, and B. Abaza. Thinning algorithms for arabic ocr. In *IEEE Pacific Rim Conference on Communications, Computers and Signal*, volume 1, pages 248–251, 1993.

[TYRW14] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, 2014.

[UHB13] A. Ul-Hasan and T.M. Breuel. Can we build language-independent ocr using lstm networks? In *International Workshop on Multilingual OCR*, pages 9:1–9:5, 2013.

[UHBAR+13] A. Ul-Hasan, S. Bin Ahmed, F. Rashid, F. Shafait, and T.M. Breuel. Offline printed urdu nastaleeq script recognition with bidirectional lstm networks. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1061–1065, 2013.

[Vin02] A. Vinciarelli. A survey on off-line cursive word recognition. *Pattern recognition*, 35(7):1433–1446, 2002.

[VJ01a] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages I–511–I–518, 2001.

[VJ01b] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511–518, 2001.

[VJ04] P. Viola and M.J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[VLBM08] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[VML94] R. Vaillant, C. Monrocq, and Y. LeCun. Original approach for the localisation of objects in images. In *IEE Proceedings - Vision, Image and Signal Processing*, volume 141, pages 245 – 250, 1994.

[VV05] A. Vezhnevets and V. Vezhnevets. Modest adaboost-teaching adaboost to generalize better. In *Graphicon*, volume 12, pages 987–997, 2005.

[WB91] I.H. Witten and T.C. Bell. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4):1085–1094, 1991.

[WB10] K. Wang and S. Belongie. *Word spotting in the wild*. Springer, 2010.

[WBB11] K. Wang, B. Babenko, and S. Belongie. End-to-end scene text recognition. In *International Conference on Computer Vision (ICCV)*, pages 1457–1464, 2011.

[WEG+10] M. Wöllmer, F. Eyben, A. Graves, B. Schuller, and G. Rigoll. Bidirectional lstm networks for context-sensitive keyword detection in a cognitive virtual agent framework. *Cognitive Computation*, 2(3):180–190, 2010.

[Wer88a] P.J. Werbos. Backpropagation: Past and future. In *IEEE International Conference on Neural Networks*, pages 343–353, 1988.

[Wer88b] P.J. Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1(4):339–356, 1988.

[WHH+89] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K.J. Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(3):328–339, 1989.

[WJ06] C. Wolf and L.M. Jolion. Object count/area graphs for the evaluation of object detection and segmentation algorithms. *International Journal on Document Analysis and Recognition*, 8(4):280–296, August 2006.

[WK03] K. Wang and J.A. Kangas. Character location in scene images from digital camera. *Pattern recognition*, 36(10):2287–2299, 2003.

[WK11] T. Wakahara and K. Kita. Binarization of color character strings in scene images using k-means clustering and support vector machines. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 274–278, 2011.

[WWCN12] T. Wang, D.J. Wu, A. Coates, and A.Y. Ng. End-to-end text recognition with convolutional neural networks. In *International Conference on Pattern Recognition (ICPR)*, pages 3304–3308, 2012.

[WYL09] Q.-F. Wang, F. Yin, and C.-L. Liu. Integrating language model in handwritten chinese text recognition. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1036–1040, 2009.

[WYY+10] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3360–3367, 2010.

[XHC+01] J. Xi, X-S. Hua, X-R. Chen, L. Wenyin, and H-J. Zhang. A video text detection and recognition system. In *International Conference on Multimedia and Expo*, pages 873–876, 2001.

[XJ07] P. Xu and F. Jelinek. Random forests and the data sparseness problem in language modeling. *Computer Speech & Language*, 21(1):105–152, 2007.

[XPD06] P. Xiu, L. Peng, and X. Ding. Multi-queue merging scheme and its application in arabic script segmentation. In *International Conference on Document Image Analysis for Libraries*, pages 6–pp, 2006.

[YBG14] S. Yousfi, S.-A. Berrani, and C. Garcia. Arabic text detection in videos using neural and boosting-based approaches: Application to video indexing. In *International Conference on Image Processing (ICIP)*, pages 3028–3032, 2014.

[YBG15a] S. Yousfi, S.-A. Berrani, and C. Garcia. Alif: A dataset for arabic embedded text recognition in tv broadcast. In *The 6th International Workshop on Camera Based Document Analysis and Recognition - ICDAR Workshop*, pages 1221–1225, 2015.

[YBG15b] S. Yousfi, S.-A. Berrani, and C. Garcia. Deep learning and recurrent connectionist-based approaches for arabic text recognition in videos. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1026–1030, 2015.

[YBL+12] C. Yao, X. Bai, W. Liu, . Ma, and Z. Tu. Detecting texts of arbitrary orientations in natural images. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1083–1090, 2012.

[YD15] Qixiang Ye and David Doermann. Text detection and recognition in imagery: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 37(7):1480–1500, 2015.

[YGH04] Q. Ye, W. Gao, and Q. Huang. Automatic text segmentation from complex background. In *International Conference on Image Processing*, volume 5, pages 2905–2908, 2004.

[YHGZ05] Q. Ye, Q. Huang, W. Gao, and D. Zhao. Fast and robust text detection in images and video frames. *Image and Vision Computing*, 23(6):565–576, 2005.

[YPX09] J. Yi, Y. Peng, and J. Xiao. Using multiple frame integration for the text recognition of video. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 71–75, 2009.

[YT11] C. Yi and Y. Tian. Text string detection from natural scenes by structure-based partition and grouping. *IEEE Transactions on Image Processing*, 20(9):2594–2605, 2011.

[YZB+13] C. Yao, X. Zhang, X. Bai, W. Liu, Y. Ma, and Z. Tu. Rotation-invariant features for multi-oriented text detection in natural images. *PloS one*, 8(8), 2013.

[ZCX+07] L. Zhang, R. Chu, S. Xiang, S. Liao, and S.Z. Li. Face detection based on multi-block lbp representation. In *Proceedings of the international conference on Advances in Biometrics*, pages 11–18, 2007.

[ZFLW15] Z. Zhao, C. Fang, Z. Lin, and Y. Wu. A robust hybrid method for text detection in natural scenes by learning-based partial differential equations. *Neurocomputing*, 168:23–34, 2015.

[ZGCZ05] H. Zhang, W. Gao, X. Chen, and D. Zhao. Learning informative features for spatial histogram-based object detection. In *International Joint Conference on Neural Networks*, volume 3, pages 1806–1811, 2005.

[Zid04] A. Zidouri. Oran: a basis for an arabic ocr system. In *International Symposium on Intelligent Multimedia, Video and Speech Processing*, pages 703–706, 2004.

[ZLT10] Z. Zhou, L. Li, and C.L. Tan. Edge based binarization for video text images. In *International Conference on Pattern Recognition (ICPR)*, pages 133–136, 2010.

[ZMFEB+14] F. Zamora-Martínez, V. Frinken, S. Espana-Boquera, M.J. Castro-Bleda, A. Fischer, and H. Bunke. Neural network language models for off-line handwriting recognition. *Pattern Recognition*, 47(4):1642–1652, 2014.

[ZSSJ05] A. Zidouri, M. Sarfraz, S.A. Shahab, and S.M. Jafri. Adaptive dissection based subword segmentation of printed arabic text. In *International Conference on Information Visualisation*, pages 239–243, 2005.

[ZYB16] Y. Zhu, C. Yao, and X. Bai. Scene text detection and recognition: Recent advances and future trends. *Frontiers of Computer Science*, 10(1):19–36, 2016.

FOLIO ADMINISTRATIF

THESE DE L'UNIVERSITE DE LYON OPEREE AU SEIN DE L'INSA LYON

NOM : YOUSFI
(avec précision du nom de jeune fille, le cas échéant)

DATE de SOUTENANCE : 06/07/2016

Prénoms : Sonia

TITRE : Embedded Arabic text detection and recognition in videos

NATURE : Doctorat

Numéro d'ordre : 2016LYSEI069

Ecole doctorale : Informatique et Mathématiques de Lyon (EDA 512)

Spécialité : Informatique

RESUME :

This thesis focuses on Arabic embedded text detection and recognition in videos. Different approaches robust to Arabic text variability (fonts, scales, sizes, etc.) as well as to environmental and acquisition condition challenges (contrasts, degradation, complex background, etc.) are proposed.
We introduce different machine learning-based solutions for robust text detection without relying on any pre-processing. The first method is based on Convolutional Neural Networks (ConvNet) while the others use a specific boosting cascade to select relevant hand-crafted text features.
For the text recognition, our methodology is segmentation-free. Text images are transformed into sequences of features using a multi-scale scanning scheme. Standing out from the dominant methodology of hand-crafted features, we propose to learn relevant text representations from data using different deep learning methods, namely Deep Auto-Encoders, ConvNets and unsupervised learning models. Each one leads to a specific OCR (Optical Character Recognition) solution. Sequence labeling is performed without any prior segmentation using a recurrent connectionist learning model. Proposed solutions are compared to other methods based on non-connectionist and hand-crafted features.
In addition, we propose to enhance the recognition results using Recurrent Neural Network-based language models that are able to capture long-range linguistic dependencies. Both OCR and language model probabilities are incorporated in a joint decoding scheme where additional hyper-parameters are introduced to boost recognition results and reduce the response time.
Given the lack of public multimedia Arabic datasets, we propose novel annotated datasets issued from Arabic videos. The OCR dataset, called ALIF, is publicly available for research purposes. As the best of our knowledge, it is the first public dataset dedicated for Arabic video OCR.
Our proposed solutions were extensively evaluated. Obtained results highlight the genericity and the efficiency of our approaches, reaching a word recognition rate of 88.63% on the ALIF dataset and outperforming well-known commercial OCR engine by more than 36%.

MOTS-CLÉS :
Arabic text detection, Arabic text recognition, video contents, Deep Learning, Auto-Encoder, Convolution Neural Networks, Boosting, Recurrent Neural Networks, Connectionist language model, Decoding

Laboratoire (s) de recherche : Laboratoire d'InfoRmatique en Image et Systèmes d'information (LIRIS)

Directeur de thèse: GARCIA, Christophe

Président de jury :

Composition du jury :

M. CANU, Stéphane	PRU, INSA Rouen	Rapporteur
M. WENDLING, Laurent	PRU, Université Paris Descartes	Rapporteur
M. THIRAN, Jean-Philippe	PRU, EPFL Lausanne	Examinateur
M. COUASNON, Bertrand	MC/HDR, INSA de Rennes	Examinateur
M. GARCIA, Christophe	PRU, INSA de Lyon	Directeur de thèse
M. BERRANI, Sid-Ahmed	Chef d'équipe/HDR, Orange Labs Rennes	Co-encadrant