Reconnaissance de l'écriture manuscrite avec des réseaux récurrents

Luc Mioulet

Normandie Université

THÈSE

Pour obtenir le grade de Docteur

opérée par l'Université de Rouen

Spécialité : Informatique

Reconnaissance de l'écriture manuscrite avec des réseaux récurrents

Présentée et soutenue publiquement par
Luc Mioulet

Thèse soutenue publiquement le 3 juillet 2015 devant le jury composé de		
Christian Viard-Gaudin	Université de Nantes	*Président*
Lionel Prevost	Université des Antilles et de la Guyane	*Rapporteur*
Bertrand Coüasnon	INSA de Rennes	*Rapporteur*
Christophe Garcia	INSA de Lyon	*Examinateur*
Thierry Paquet	Université de Rouen	*Directeur*
Clément Chatelain	INSA de Rouen	*Encadrant*
Stéphan Brunessaux	Airbus DS	*Invité*

Thèse dirigée par Thierry Paquet, Laboratoire d'Informatique, de Traitement de l'Information et des Systèmes

litis

UNIVERSITÉ DE ROUEN

Résumé

La numérisation massive de documents papier a fait apparaître le besoin d'avoir des systèmes de reconnaissance de l'écriture extrêmement performants. La numérisation de ces documents permet d'effectuer des opérations telles que des recherches de mots clefs ou l'extraction d'informations de haut niveau (titre, auteur, adresses, et.). Cependant la reconnaissance de l'écriture et en particulier l'écriture manuscrite ne sont pas encore au niveau de performance de l'homme sur des documents complexes, ce qui restreint ou nuit à certaines applications. Cette thèse CIFRE entre Airbus DS et le LITIS, dans le cadre du projet MAURDOR, a pour but de mettre en avant et d'améliorer les méthodes état de l'art dans le domaine de la reconnaissance de l'écriture manuscrite.

Nos travaux comparent différents systèmes permettant d'effectuer la reconnaissance de l'écriture manuscrite. Nous comparons en particulier différentes caractéristiques et différents classifieurs dynamiques : i) Modèles de Markov Cachés (MMC), ii) hybride réseaux de neurones/MMC, iii) hybride réseaux récurrents « Bidirectional Long Short Term Memory - Connectionist Temporal Classification » (BLSTM-CTC)/MMC et iv) hybride Champs Aléatoires Conditionnels (CAC)/MMC. Les comparaisons sont réalisées dans les conditions de la tâche WR2 de la compétition ICDAR 2009, c'est à dire une tâche de reconnaissance de mots isolés avec un dictionnaire de 1600 mots. Nous montrons la supériorité de l'hybride BLSTM-CTC/MMC sur les autres classifieurs dynamiques ainsi que la complémentarité des sorties des BLSTM-CTC utilisant différentes caractéristiques.

Notre seconde contribution vise à exploiter ces complémentarités. Nous explorons des stratégies de combinaisons opérant à différents niveaux de la structure des BLSTM-CTC : bas niveau (en entrée), moyen niveau (dans le réseau), haut niveau (en sortie). Nous nous plaçons de nouveau dans les conditions de la tâche WR2 de la compétition ICDAR 2009. De manière générale nos combinaisons améliorent les résultats par rapport aux systèmes individuels, et nous avoisinons les performances du meilleur système de la compétition. Nous avons observé que certaines combinaisons sont adaptées à des systèmes sans lexique tandis que d'autres sont plus appropriées pour des systèmes avec lexique.

Notre troisième contribution se situe sur deux applications liées à la re-

connaissance de l'écriture. Nous présentons un système de reconnaissance de la langue ainsi qu'un système de détection de mots clefs, à partir de requêtes images et de requêtes de texte. Dans ces deux applications nous présentons une approche originale faisant appel à la reconnaissance de l'écriture. En effet la plupart des systèmes de la littérature extraient des caractéristiques des image pour déterminer une langue ou trouver des images similaires, ce qui n'est pas nécessairement l'approche la plus adaptée au problème à traiter. Nos approches se basent sur une phase de reconnaissance de l'écriture puis une analyse du texte afin de déterminer la langue ou de détecter un mot clef recherché.

Mots clefs : analyse d'images de documents, reconnaissance de l'écriture, réseaux récurrents, Modèles de Markov Cachés, champs aléatoires Conditionnels, reconnaissance de langue, détection de mots clefs.

Abstract

Mass digitization of paper documents requires highly efficient optical character recognition systems. Digital versions of paper documents enable the use of search engines through keyword dectection or the extraction of high level information (e.g. : titles, author, dates). Unfortunately writing recognition systems and especially handwriting recognition systems are still far from having similar performance to that of a human being on the most difficult documents. This industrial PhD (CIFRE) between Airbus DS and the LITIS, that took place within the MAURDOR project time frame, aims to seek out and improve the state of the art systems for handwriting recognition.

We compare different systems for handwriting recognition. Our comparisons include various feature sets as well as various dynamic classifiers : i) Hidden Markov Models, ii) hybrid neural network/HMM, iii) hybrid recurrent network Bidirectional Long Short Term Memory - Connectionist Temporal Classification (BLSTM-CTC)/MMC, iv) a hybrid Conditional Random Fields (CRF)/HMM. We compared these results within the framework of the WR2 task of the ICDAR 2009 competition, namely a word recognition task using a 1600 word lexicon. Our results rank the BLSTM-CTC/HMM system as the most performant, as well as clearly showing that BLSTM-CTCs trained on different features are complementary.

Our second contribution aims at using this complementarity. We explore various combination strategies that take place at different levels of the BLSTM-CTC architecture : low level (early fusion), mid level (within the network), high level (late integration). Here again we measure the performances of the WR2 task of the ICDAR 2009 competition. Overall our results show that our different combination strategies improve on the single feature systems, moreover our best combination results are close to that of the state of the art system on the same task. On top of that we have observed that some of our combinations are more adapted for systems using a lexicon to correct a mistake, while other are better suited for systems with no lexicon.

Our third contribution is focused on tasks related to handwriting recognition. We present two systems, one designed for language recognition, the other one for keyword detection, either from a text query or an image query.

For these two tasks our systems stand out from the litterature since they use a handwriting recognition step. Indeed most litterature systems focus on extracting image features for classification or comparison, wich does not seem approriate given the tasks. Our systems use a handwriting recognition step followed either by a language detection step or a word detection step, depending on the application.

Keywords : document image analysis, handwriting recognition, recurrent neural network, hidden Markov models, conditional random fields, language detection, keyword spotting.

Remerciements

Je tiens à remercier Thierry Paquet, Clément Chatelain et Stéphan Brunessaux de m'avoir permis d'effectuer cette thèse CIFRE. La qualité de leur encadrement, leur présence continue, et leur soutien m'ont aidé à mener à bien ce projet de trois ans.

Mes remerciements vont aux rapporteurs de cette thèse Christian Viard-Gaudin et Lionel Prévost pour avoir accepté de relire et juger mes travaux.

Je tiens à remercier l'ensemble de l'équipe TCOIC 4 et en particulier Sylvie Brunessaux, Bruno Grilhères, Patrick Giroux, Yann Monbrun, Émilien Bondu, Véronique Armand, Esther Nicart, Romain Noël, Jennifer Renoux, Nicolas Le Guillarme avec qui j'ai eu l'occasion de travailler et d'échanger durant mes trois années sur le site Airbus DS de Val de Reuil.

Je remercie l'ensemble des membres du LITIS avec qui j'ai échangé sur différents sujets scientifiques et qui m'ont apporté leurs connaissances dans différents domaines. Je remercie Kamel Ait-Mohand pour son apport technique concernant les Modèles de Markov Cachés. Je remercie Utpal Garain de l'université de Calcutta. Il a eu la gentillesse de nous fournir des données d'apprentissage pour le bangla et l'assamais et nous a permis de comprendre les bases de ces scripts.

Je tiens à remercier en particulier Gautier Bideault et Philippine Barlas qui m'ont beaucoup aidé durant ma thèse et avec qui j'ai partagé d'excellents moments durant ces trois années de thèse et ans d'INSA. Vous avoir comme collègues de TP et amis a été une vraie chance pour moi ! Je pense que nos bureaux n'ont jamais été aussi parfumés !

J'exprime ma gratitude à Nicolas Oliveira et Julien Pladeau qui ont relu ce manuscrit afin d'en améliorer sa lecture. Merci à vous pour ces 2000 et des patates victoires sur League, les 1500 heures sur GW2 et les milliers d'heures passées en votre joviale compagnie !

Je remercie mes parents et ma famille qui ont toujours été à mes côtés et qui m'ont apporté leur soutien tout au long de mes études. Je n'aurai jamais pu arriver ici sans vous.

Je remercie Gérard Devillers mon professeur de Karaté et ami, qui m'a aidé à me dépasser et à évoluer, dans ma vie comme dans le Karaté. Je remer-

cie aussi l'ensemble des membres du Karaté Club Sottevillais : Lydia, Baptiste, Éric, Séverine, Yves, Jean-Pierre, ainsi que tous les gradés et non gradés, adultes comme enfants ! Arigato gozaimasu !

Merci à Émeline pour ces deux années durant lesquelles on a partagé nos vies entre les voyages, le Karaté et les oiseaux !

Je remercie l'ensemble des personnes qui sont venus assister et me soutenir durant ma soutenance de thèse.

Table des matières

Chapitre 1

Introduction générale

Les premières traces d'écriture nous proviennent du IV^{e} millénaire av. J.-C de Mésopotamie. Le développement du commerce dans le berceau des civilisations, et la nécessité de communiquer sur des distances de plus en plus importantes des messages de plus en plus complexes, ont poussé les mésopotamiens à développer un système utilisant des symboles représentant des quantités, des phonèmes, des idées afin d'inscrire sur différents supports une mémoire des informations. L'écriture permet à la fois de communiquer sur des grandes distances, mais aussi à travers le temps. Sans l'écriture nous n'aurions pas accès au savoir des hommes comme Platon, Socrate, Galilée ou Newton et notre monde serait bien différent.

À l'ère numérique, les documents manuscrits (et plus généralement les documents papier) semblent jouer un rôle de moins en moins important. Nous privilégions de plus en plus les supports numériques au détriment du papier pour inscrire et partager notre savoir. Il y a quinze ans nous utilisions encore des calepins pour noter les numéros de téléphone de nos amis, nos recettes, nos notes de réunions ou notre agenda. Aujourd'hui toutes ces activités peuvent être réalisées sur des supports numériques qui facilitent le partage de ces contenus avec d'autres personnes. Cependant l'écriture manuscrite bien que progressivement remplacée par des applications numériques reste cependant extrêmement présente dans notre monde. Nous continuons d'écrire nos chèques, nos adresses postales, nos cartes postales, nous remplissons nos formulaires, nous posons nos idées sur des feuilles de brouillon.

La masse des documents papier continue de croître, et de plus en plus d'industries ou de services ont un besoin de numériser ces documents papier. En effet une fois numérisés ces documents permettent d'automatiser des processus, comme l'extraction automatique d'informations de formulaires. Les logiciels capables de numériser des documents sont appelés des systèmes de Reconnaissance Optique de Caractères (ROC). La numérisation de documents permet aussi leur exploitation sans dégrader l'original, comme dans le cas de documents anciens. Malheureusement la numérisation de ces documents, et en

particulier l'extraction du contenu texte de ces documents, n'est pas parfaite. Bien que les ROC développés soient de plus en plus performants aucun ne surpasse ni n'égale un opérateur humain au niveau du taux de reconnaissance. Des chercheurs ont développé des stratégies de corrections d'erreurs, comme l'utilisation de lexiques ou des règles grammaticales, permettant d'améliorer les résultats. Ces améliorations permettent d'obtenir des applications industrielles avec un faible taux d'erreur : des millions de lettres sont ainsi acheminées automatiquement à l'aide de ROC, la majorité des chèques sont eux aussi traités entièrement par des ROC.

Ces ROC sont néanmoins limités à des documents très spécifiques avec un lexique très restreint. Il n'existe pas à ce jour de ROC capable de traiter des documents hétérogènes, tout en offrant des performances utilisables dans un contexte industriel. Notre étude s'inscrit dans le cas du projet d'étude amont Moyens AUtomatiques de Reconnaissance de DOcuments écRits (MAURDOR), qui a pour but de permettre une évaluation de l'état de l'art des systèmes ROC capables de traiter des documents de sources hétérogènes. Cette thèse CIFRE menée en collaboration entre Airbus DS et le LITIS a pour but de mettre en avant les techniques modernes de reconnaissance de l'écriture, et en particulier pour l'écriture manuscrite, et de les faire progresser.

La reconnaissance de l'écriture manuscrite est un domaine scientifique dans lequel il existe de nombreux systèmes ayant déjà fait leurs preuves. De manière générale ces systèmes transforment, à l'aide d'algorithmes de traitement d'images, une image en une séquence de vecteurs de données. Cette séquence de vecteurs de données peut ensuite être traitée par une méthode statistique de classification de séquences, aussi appelée un classifieur dynamique. Dans notre étude nous comparons différents systèmes basés sur différents classifieurs dynamiques. Nous présentons nos contributions sur les améliorations du classifieur dynamique le plus performant de notre comparaison. Finalement nous présentons un certain nombre d'applications rendues possibles par les travaux récents dans le domaine.

Le chapitre 2 introduit un ensemble de classifieurs dynamique fréquemment utilisés dans le domaine de la reconnaissance de l'écriture. Nous nous intéressons en particulier aux classifieurs dynamiques capables de traiter des séquences, nous présentons aussi des classifieurs statiques, qui associés à des classifieurs dynamiques peuvent aussi être utilisés pour la classification de séquences.

Le chapitre 3 présente dans un premier temps un ensemble de différents

algorithmes utilisés régulièrement dans le domaine de la reconnaissance de l'écriture. Ces algorithmes permettent de simplifier une image, d'extraire des informations d'une image ou bien de corriger des erreurs de reconnaissance effectuées par un classifieur dynamique. Dans un second temps nous présentons plusieurs familles de systèmes de reconnaissance de l'écriture ainsi que des systèmes issus de ces familles afin de mettre en avant leurs avantages et leurs inconvénients.

Le chapitre 4 présente une comparaison entre ces différentes familles de systèmes sur une même base. Nous présentons en particulier deux systèmes exploratoires que nous comparons à des systèmes de références dans le domaine de la reconnaissance de l'écriture manuscrite. Nous présentons une chaîne de reconnaissance effectuant la reconnaissance de l'écriture. Entre deux systèmes nous ne changeons que le classifieur dynamique utilisé, nous comparons ainsi un ensemble de classifieurs dynamiques dans le même contexte dans le but de déterminer le classifieur le plus pertinent dans ce contexte. De plus nous explorons l'impact de différentes caractéristiques sur ces systèmes.

Le chapitre 5 explore différentes stratégies de combinaisons pour des réseaux récurrents (l'un de nos classifieurs dynamiques). Ces stratégies se basent sur la complémentarité des sorties des réseaux de neurones appris avec différentes caractéristiques. Notre contribution permet de choisir la combinaison la plus efficace en fonction des besoins de l'application finale.

Le chapitre 6 présente deux applications de la reconnaissance de l'écriture manuscrite. Appliquer la ROC à un document n'est pas nécessairement une fin en soit, il peut être intéressant d'en extraire de l'information, comme les langues du document ou des mots clefs. Cependant les sorties des systèmes de ROC ne sont pas parfait, il est donc nécessaire de mesurer l'impact de ces imperfections sur différentes problématiques élémentaires qui pourraient paraître triviales si on les implémentait avec des données parfaites, mais qui présentent néanmoins un intérêt primordial pour la mise en œuvre des chaînes de lecture automatiques d'images de documents manuscrits.

Chapitre 2

Classification statistiques

Reconnaître un objet c'est lui assigner une étiquette, un label, une classe. La reconnaissance de l'écriture est donc un problème de classification, pour une image d'un mot ou d'une phrase nous désirons déterminer l'ensemble des caractères contenus dans cette image. Nous nous intéressons principalement à la reconnaissance de caractères non isolés, la reconnaissance de caractères isolés possède un champ applicatif très restreint (contenu extrait de peignes des formulaires). La reconnaissance de mots ou de phrases possède un champ applicatif bien plus large (lettres manuscrites, formulaires, annotations, etc.), cependant elle est aussi plus complexe. En effet dans un mot ou dans une phrase il existe des dépendances temporelles entre les différents caractères, leur séquence n'est pas aléatoire, il est donc important de modéliser ces dépendances au moment de la reconnaissance. L'utilisation de méthodes statistiques adaptées au problème de la reconnaissance de l'écriture est donc prépondérant.

Dans ce chapitre nous présentons un ensemble non exhaustif de méthodes statistiques de classification, ou plus communément des classifieurs. Dans l'ensemble de ce chapitre nous conservons les notations suivantes par soucis de cohérence :

- les entrées du système : $X = \{x_1, x_2, \ldots, x_t, \ldots, x_T\}$, une séquence de vecteurs de données (ou vecteur d'observations) x_t de longueur T. Un vecteur de données $x_t = (x_t^1, x_t^2, \ldots, x_t^N)^\intercal$ est composé d'un nombre N de données. Ces données appartiennent au domaine $\mathbb{R}$.
- les classes : $C = \{c_1, c_2, \ldots, c_k, \ldots, c_K\}$, l'ensemble des K classes parmi lesquelles le classifieur doit effectuer sa décision.
- les sorties du classifieur : $Y = \{y_1, y_2, \ldots, y_t, \ldots, y_T\}$, une séquence de longueur L de vecteurs de sorties du classifieur $y_t = (y_t^1, y_t^2, \ldots, y_t^K)^\intercal$ représentant les décisions probabilisées du classifieur.

Les classifieurs peuvent être catégorisés selon leur dynamique de classification en deux catégories : les classifieurs statiques pour qui les vecteurs dans une même séquence sont classifiés indépendamment les uns des autres et les classifieurs dynamiques pour qui les vecteurs sont classifiés en s'appuyant sur un

contexte temporel (par exemple la classification précédente dans la séquence, ou bien sur le contenu d'une mémoire). Nous présentons dans un premier temps les classifieurs statiques, puis nous présentons les classifieurs dynamiques.

2.1 Classifieurs statiques

Les classifieurs statiques regroupent toutes les méthodes statistiques de classification sans mémoire permettant d'assigner une représentation numérique (un vecteur de données) à une classe. Dans le cas des classifieurs statiques les liens temporels entre les données, ou les sorties, sont inexistants. Nous considérons donc que $T = 1$. Il existe un nombre important de classifieurs statiques : réseaux de neurones [150], réseaux de neurones profonds [15], Séparateurs à Vastes Marges [45] (SVM), K-plus proche voisins [39], forêts aléatoires[31], etc. . Les SVM, qui sont des méthodes à noyaux maximisant la séparation des classes, et les forêts aléatoires, qui sont des ensembles d'arbres de décisions minimisant le sur-apprentissage, ne sont pas présentés dans cette section car ils sont très peu utilisés dans le domaine de la reconnaissance de l'écriture. Nous présentons d'abord les réseaux de neurones puis les réseaux de neurones profonds.

2.1.1 Réseaux de neurones

Les réseaux de neurones [23, 150] sont décrits par des assemblages de neurones interconnectés entre eux. Il existe de nombreuses architectures de réseaux de neurones, nous décrivons ici le perceptron multi-couches, qui est une architecture structurant les neurones en couches de neurones. C'est une architecture très commune afin de réaliser une classification [28, 29, 46, 134]. Nous présentons cette architecture puis nous présentons des méthodes d'apprentissage permettant d'optimiser un perceptron multi-couches.

2.1.1.1 Perceptron multi-couches

Dans un perceptron multi-couches, un neurone d'une couche est connecté en entrée à tous les neurones de la couche précédente, et en sortie aux neurones de la couche suivante (voir figure 2.1 et sa version simplifiée 2.2). On distingue trois nomenclatures de couches : la couche d'entrée E, les couches cachées H_s et la couche de sortie S. La couche d'entrée n'est pas réellement une couche, il s'agit du vecteur de données, elle n'est pas comptée dans le nombre de couches

d'un réseau de neurones bien que souvent représentée dans les illustrations. La couche de sortie réalise l'association entre les classes de l'entrée et la représentation produite par les couches cachées, en effet chaque neurone sur la couche de sortie représente une classe. Les couches cachées, relient la couche d'entrée à la couche de sortie, elles réalisent un ensemble de transformations des données d'entrée permettant à la couche de sortie d'effectuer une décision.

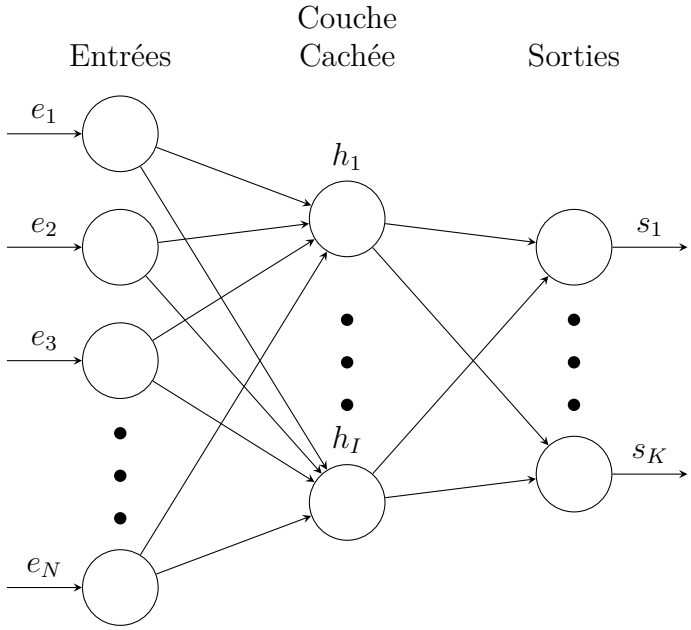


FIGURE 2.1 – Perceptron multi-couches

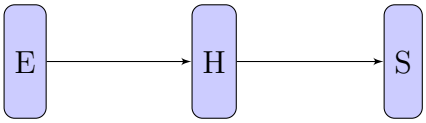


FIGURE 2.2 – Perceptron multi-couches, schéma simplifié

Soit un neurone j sur la couche $h-1$ et un neurone n sur la couche h. Ces deux neurones sont connectés par un lien La connexion du neurone j au neurone n est pondérée, cette pondération w_{nj} permet au réseau de neurones d'être optimisé pour une classification spécifique. Un neurone de la couche h est donc connecté à tous les neurones de la couche précédente $h-1$. Chaque neurone est décrit par une fonction de transfert $f(s_n^h)$ (ou fonction d'activation), qui est généralement une sigmoïde, avec z_j^{h-1} la sortie d'un neurone de la couche précédente et $s_n^h = \sum_{j=1}^{J}(z_j^{h-1} w_{nj})$ la somme pondérée des entrées. La

fonction de transfert s'écrit donc :

$$f(s_n^h) = f\left(\sum_{j=1}^{J} z_j^{h-1} w_{nj}\right) \tag{2.1}$$

La figure 2.3 illustre un neurone isolé et ses connexions aux neurones de la couche précédente. Une couche est donc un ensemble de neurones qui possèdent généralement la même fonction de transfert et qui sont pleinement interconnectés à la couche précédente et suivante. L'utilisation d'une fonction de transfert non linéaire permet au réseau d'approximer des fonctions non linéaires, et donc de répondre à un problème de classification avec des frontières non linéaires dans des espaces de grande dimension. Il est cependant nécessaire d'optimiser un réseau de neurones à l'aide d'un algorithme d'apprentissage, nous présentons dans la suite quelques uns de ces algorithmes.

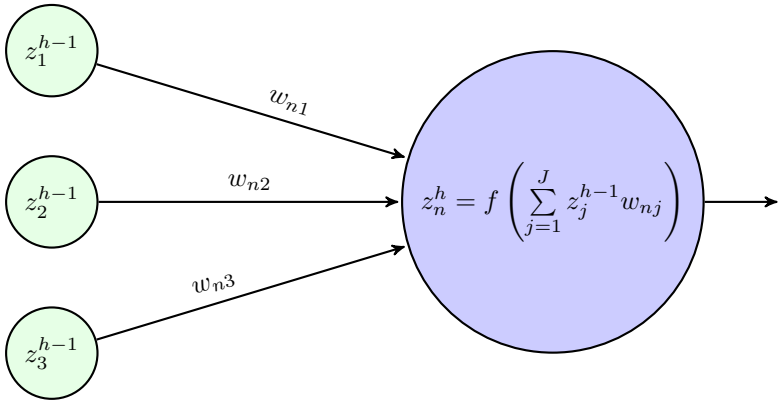


FIGURE 2.3 – Neurone artificiel

2.1.1.2 Apprentissage

L'optimisation d'un réseau de neurones par rapport à une tâche se fait par optimisation des poids du réseau. L'objectif est de minimiser la différence entre les sorties idéales y et les sorties produites par le réseau o, c'est à dire de minimiser l'erreur du réseau sur une base $\mathcal{D} = (\{x_1, y_1\}, \{x_2, y_2\}, \ldots \{x_p, y_p\} \ldots \{x_P, y_P\})$ avec P le nombre d'éléments dans la base. L'erreur est définie comme :

$$Err = \frac{1}{2} \sum_{i=1}^{P} ||y_p - o_p||^2 \tag{2.2}$$

La rétro-propagation du gradient de l'erreur de sortie [112] est la méthode la plus répandue pour l'apprentissage, elle permet de déterminer un minimum local pour la fonction d'erreur. Il s'agit d'une méthode itérative de descente

de gradient, dont le critère d'arrêt est généralement un nombre d'itérations ou un arrêt de la progression vers la solution. Le réseau est initialisé avec des poids définis aléatoirement, ou définis de manière pseudo-aléatoire [180]. Durant une itération chaque exemple d'apprentissage x_p est propagé dans le réseau de neurones, ce qui produit donc la sortie du réseau o_p. L'erreur entre la sortie obtenue d'un neurone de la couche de sortie o_p et la sortie désirée y_p est alors mesurée :

$$err_k = \frac{1}{2}(o_{pk} - y_{pk})^2 \tag{2.3}$$

où o_{pk} sont les sorties obtenues en sortie du neurone k pour l'élément d'apprentissage p et y_{pk} sont les sorties obtenues en sortie du neurone k pour l'élément d'apprentissage p. Cette erreur est rétro-propagée de la couche de sortie S vers la couche d'entrée E, ainsi chaque pondération est affectée de l'erreur qu'elle a effectuée pour l'élément d'apprentissage p. La rétro-propagation s'effectue en minimisant l'erreur par une méthode de descente de gradient, il est donc nécessaire de calculer le gradient de l'erreur pour chaque pondération w_l (avec l un couple) dans le réseau :

$$\nabla E = \left(\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \dots \frac{\partial E}{\partial w_l} \right) \tag{2.4}$$

les dérivées partielles de l'erreur par rapport à chaque poids doivent donc être calculée en dérivant la fonction de transfert des neurones. Les pondérations des connexions sont ensuite modifiées pour réduire cette erreur :

$$\Delta w_i = -\mu \frac{\partial E}{\partial w_i} \tag{2.5}$$

avec μ représentant une constante d'apprentissage, précisant l'impact de la correction des erreurs sur le réseau. Ce paramètre n'est pas anodin, réglé avec un pas trop petit ou trop grand il peut empêcher l'erreur d'atteindre un minimum local pertinent. Il est généralement conseillé [113] de l'initialiser avec une valeur « importante » puis de le faire décroître après chaque itération sur la base d'apprentissage.

Différentes variantes de rétro-propagation existent comme la rétro-propagation par lot [11] (qui met à jour les poids après qu'un certain nombre d'éléments d'apprentissages aient été présentés) ou la RProp [148] (qui prend en compte l'évolution du signe de la dérivée partielle de chaque pondération entre deux itérations). Ces méthodes, dites du premier ordre, ne sont pas les plus performantes car seule la direction générale de l'optimisation est estimée à chaque

itération. Des méthodes du second ordre existent, permettant d'estimer également le pas optimal à chaque itération, mais elles sont plus complexes et plus gourmandes en ressources [11, 124].

L'empilement des couches d'un réseau de neurones permet de décrire des frontières de décision complexes, ce qui est intéressant dans le cas de la reconnaissance de l'écriture. Les réseaux de neurones ont aussi l'avantage de supporter de très hautes dimensions en entrée comme en sortie. Leur inconvénient majeur est le nombre d'hyper-paramètres qui les définissent. Comme nous venons de l'expliquer, un réseau de neurones est défini par de multiples hyper-paramètres : nombre de couches, nombre de neurones dans chaque couche, fonction de transfert de chaque couche, paramètres de la rétro-propagation du gradient, initialisation des pondérations des connexions. Il n'existe que peu d'heuristiques pour régler ces paramètres [113, 180] et le meilleur réseau pour une tâche doit donc être découvert par l'expérimentation, après de multiples essais et erreurs.

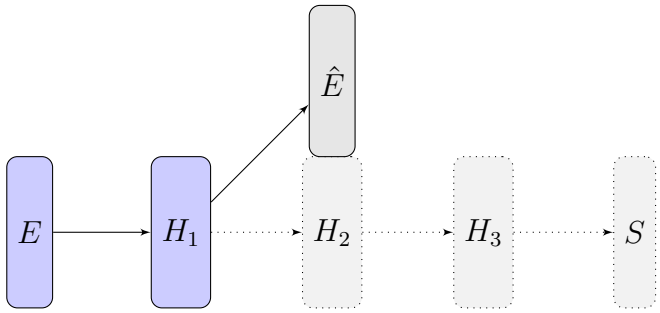
Les réseaux de neurones permettent de classifier des vecteurs de données de grandes tailles, néanmoins des vecteurs de très grandes tailles requièrent plusieurs couches de neurones pour extraire une abstraction suffisante de l'information.

Cependant, il est important de comprendre que la rétro-propagation du gradient classique est moins performante lorsqu'on ajoute des couches. En effet, l'algorithme est tel que plus une couche est éloignée de la couche de sortie, moins les poids de ses connexions seront modifiés durant la dernière étape de la rétro-propagation du gradient, ce phénomène est souvent désigné sous le terme de la disparition du gradient [17]. Cette disparition de gradient empêche les réseaux de neurones traditionnels de traiter des vecteurs de données avec de très grandes dimensions. Nous présentons maintenant une évolution des réseaux de neurones qui évite ce problème.

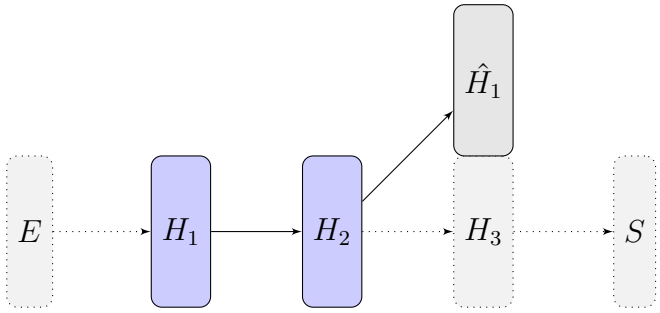
2.1.2 Réseaux de neurones profonds

Les réseaux de neurones profonds sont simplement des perceptrons multi-couches comportant plus de 3 couches. Plus un problème est complexe plus il semble naturel de rajouter de couches à un réseau.

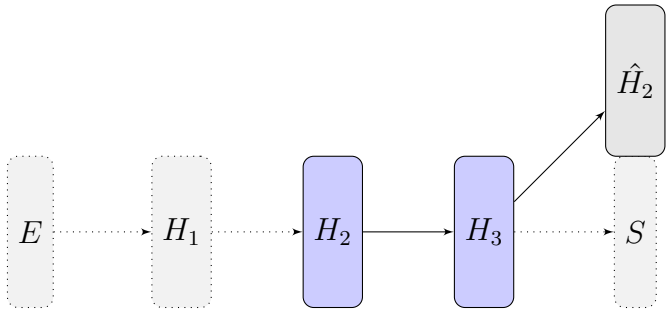
Une solution proposée au problème de la disparition du gradient [16, 92] est l'utilisation d'un apprentissage non supervisé pour apprendre les couches basses, puis un apprentissage supervisé pour les couches hautes. Les réseaux auto-encodeurs et les Restricted Blotzman Machines [63] sont des réseaux



(a) Etape 1 : apprendre les poids de la couche H_1 en reproduisant E



(b) Etape 2 : apprendre les poids de la couche H_2 en reproduisant les sorties H_1



(c) Etape 3 : apprendre les poids de la couche H_3 en reproduisant les sorties H_2

FIGURE 2.4 – Entraînement d'un réseau profond en 3 étapes

construits de cette manière. Les pondérations de chaque couche sont apprises séparément et chaque entraînement a pour objectif de reproduire en sortie ce qui lui est présenté en entrée (voir figure 2.4). On appelle ces réseaux des auto-encodeurs : les premières couches encodent l'information d'entrée dans des espaces de représentations successifs de dimensionnalité variable.

Une fois les couches cachées apprises, un dernier réglage des poids est effectué en prenant en compte la tâche à effectuer, on passe donc d'un apprentissage non supervisé à un apprentissage supervisé. C'est à dire que la dernière couche est entraînée par rapport à un objectif de classification (reconnaissance de caractères ou de symboles). Les poids proches de la couche de sortie sont alors fortement modifiés tandis que les poids éloignés de la sortie le sont faiblement.

Les réseaux profonds permettent donc de gérer des vecteurs de données

de très grandes dimensions, chaque couche générant une abstraction de cette information avant qu'elle ne serve à la tâche de classification. Ils sont utilisés pour la reconnaissance de l'écriture dans [176, 177]. Ils n'en restent pas moins des classifieurs statiques incapables de prendre en compte le contexte temporel. Tout comme les réseaux de neurones, ils n'ont pas de mémoire interne. La reconnaissance de l'écriture étant un problème avec des dépendances temporelles nous nous intéressons donc maintenant aux classifieurs dynamiques.

2.2 Classifieurs dynamiques

On rassemble sous ce terme des classifieurs possédant une mémoire interne, ou une utilisation du contexte (bas niveau sur les données, ou haut niveau sur les sorties), leur permettant de traiter des séquences de vecteur de données. Cette mémoire interne est utile dans le cas de la reconnaissance de l'écriture car il s'agit d'un signal comportant des dépendances temporelles.

La notion de segmentation des données est importante dans le cas des classifieurs dynamiques. Par segmentation nous désignons le fait d'affecter à chaque x_t un y_t correspondant (une étiquette). L'ensemble des classifieurs dynamiques effectue la segmentation lors de la décision. En revanche lors de l'apprentissage la segmentation n'est pas nécessairement existante, c'est généralement le cas dans la reconnaissance de l'écriture : un vecteur de données de longueur T représente une image représentant un mot ou une phrase G de longueur Γ, mais le vecteur des classes y_t est inconnu. Il est donc nécessaire de générer à partir de G une séquence de labels Y, c'est à dire de réaliser un étiquetage des x_t, on parle généralement d'alignement forcé. Les classifieurs dynamiques n'ont pas tous la capacité d'apprendre à segmenter au cours de l'apprentissage. Dans ce cas c'est soit à un opérateur humain, soit un autre système de reconnaissance qui est chargé de réaliser cet étiquetage en amont de la phase d'apprentissage proprement dite.

Nous présentons dans cette section les classifieurs dynamiques suivants : les Modèles de Markov Cachés (MMC), les Champs Aléatoires Conditionnels (CAC) et les réseaux récurrents.

2.2.1 Modèles de Markov Cachés

Les MMC [144] sont des modèles graphiques probabilistes génératifs, qui modélisent une séquence en intégrant des connaissances a priori : modèle de langage, lexique [34]. Les MMC sont une généralisation des modèles bayésien

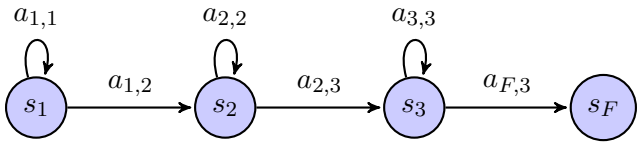


FIGURE 2.5 – MMC gauche-droite avec auto-transitions

naïf sur des données séquentielles. Nous présentons d'abord le modèle avant de présenter la modélisation et les inférences qu'il permet de réaliser.

2.2.1.1 Définition du modèle

Un MMC est une machine à états qui définit un processus doublement stochastique : un premier processus stochastique permet d'observer les états du processus, les observations $x_t \in \{v_1, v_2, \dots v_M\}$, à chaque instant, un second processus permet de modéliser les dépendances temporelles entre les états non observés, les états $y_t \in \{s_1, s_2, \dots s_K\}$, c'est le processus caché qui modélise les dépendances temporelles markoviennes, d'où la désignation de processus Marovien caché.

La probabilité de transition d'un état s_i à l'état s_j est défini comme étant la probabilité :

$$a_{ij} = p(y_t = s_j | y_{t-1} = s_i) \tag{2.6}$$

Sous une forme matricielle on note :

$$A = \begin{pmatrix} a_{11} = p(s_1, s_1) & a_{2,1} = p(s_2, s_1) & \cdots & a_{1K} = p(s_K, s_1) \\ \vdots & \vdots & \ddots & \vdots \\ a_{K1} = p(s_1, s_K) & a_{K2} = p(s_2, s_K) & \cdots & a_{KK} = p(s_K, s_K) \end{pmatrix} \tag{2.7}$$

La figure 2.5 illustre, dans le cas d'un MMC gauche-droite, les transitions des états cachés. Nous présentons ce modèle car dans le cas de la reconnaissance d'écriture ce type de modèle gauche-droite est une référence [14].

L'état initial est régi par les probabilités initiales $\pi_i = p(y_1 = s_i)$, ou sous une forme matricielle :

$$\Pi = \begin{pmatrix} \pi_1 = p(y_1 = s_1) \\ p(y_1 = s_2) \\ \vdots \\ p(y_1 = s_K) \end{pmatrix} \tag{2.8}$$

Les MMC sont des modèles génératifs, ils modélisent la probabilité d'émis-

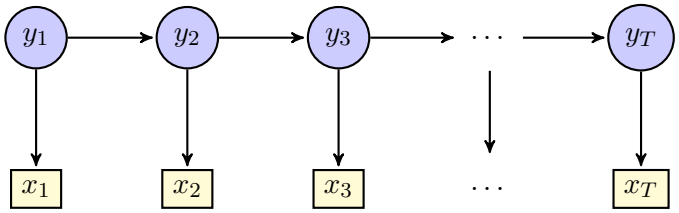


FIGURE 2.6 – Lien entre les observations x_t et les états cachés y_t dans un MMC

sion d'une observation v_m par un état donné s_k, cette probabilité est définie par la probabilité conditionnelle $b_{mk} = p(v_m|s_k)$, ou sous une forme matricielle :

$$B = \begin{pmatrix} b_{11} = p(v_1, s_1) & v_{2,1} = p(v_1|s_2) & \cdots & b_{1M} = p(v_1|s_M) \\ \vdots & \vdots & \ddots & \vdots \\ b_{K1} = p(v_K, s_1) & b_{K2} = p(v_K|s_2) & \cdots & b_{MK} = p(v_M|s_K) \end{pmatrix} \tag{2.9}$$

On distingue deux types d'observations :

— des observations discrètes, définies sur un alphabet $V = \{v_1, v_2, \ldots v_M\}$ de cardinalité M ;

— des observations continues, définies dans $\mathbb{R}^N$, où N est la dimension d'un vecteur d'observations.

La figure 2.6 illustre le lien entre les états et les observations au cours du temps. Le graphe est orienté des états vers les observations, car il s'agit d'un modèle génératif.

Les matrices A, B et Π forment un modèle $M = \{A, B, \Pi\}$.

Dans le cas d'un modèle continu, les probabilités $p(x_t = v_m|y_t = s_k)$ sont modélisées par des mélanges de Gaussiennes [21, 147, 166]. C'est ce qu'on nomme le modèle d'attache aux données. Le modèle de transition des états est appelé modèle des solutions recherchées. Après avoir présenté les bases mathématiques définissant un MMC nous présentons maintenant les outils permettant de modéliser et d'inférer différentes informations à l'aide d'un MMC.

2.2.1.2 Modélisation et inférence

Afin d'inférer ou de modéliser différentes informations différents algorithmes sont utilisés :

Forward : étant donnée une séquence partielle d'observations $X = \{x_1 \ldots x_t\}$ et le modèle M, déterminer la probabilité d'observation $p(X, y_t = s_j|M)$. Il s'agit ici de calculer la probabilité d'émission d'une séquence qui aboutit à l'état s_j à l'instant t.

Backward : étant donnée une séquence partielle d'observations $X = \{x_{t+1} \dots x_T, \}$ et le modèle M, déterminer la probabilité d'observation $p(X, y_t = s_j|M)$. Il s'agit ici de calculer la probabilité d'émission d'une séquence débutée à l'état s_j à l'instant $t + 1$.

Forward-backward : étant donnée une séquence d'observations $X = \{x_1 \dots x_T\}$ et le modèle M, déterminer la probabilité d'observation $p(X, y_t = s_j|M)$. Il s'agit ici de calculer la probabilité d'émission d'une séquence.

Viterbi : étant donnée une séquence d'observations $X = \{x_1 \dots x_T\}$ et le modèle M, déterminer la séquence d'états S ayant produit X avec la plus grande vraisemblance.

Baum-Welch : optimisation des paramètres du modèle M pour maximiser la probabilité d'une séquence d'observation X. Il s'agit ici d'apprendre un MMC.

Nous décrivons brièvement le fonctionnement de chaque algorithme.

Algorithme forward L'algorithme forward [144] permet d'inférer la probabilité de produire la séquence partielle d'observations $X_{1:t}$ et d'atteindre l'état j à l'instant t. Un calcul par récurrence est effectué :

$$\alpha_t(j) = p(x_1, \dots x_t, y_t = s_j) = (\sum_i \alpha_{t-1}(i) a_{ij}) b_j(x_t) \tag{2.10}$$

Avec l'initialisation :

$$\alpha_1(i) = \pi_i b_i(x_1) \text{ avec} \quad 1 \leq i \leq N \tag{2.11}$$

Le calcul de la variable forward peut être représenté comme le parcours d'un treillis, comme illustré sur la figure 2.7.

Algorithme backward De manière similaire à la variable forward, la variable backward permet d'inférer la probabilité de produire la séquence d'observation partielle $X_{t+1:T}$ et d'atteindre l'état i à l'instant t. Un calcul par récurrence est effectué :

$$\beta_t(j) = p(x_{t+1}, \dots x_T, y_t = s_j) = (\sum_i \beta_{t+1}(i) a_{ji}) b_i(x_{t+1}) \tag{2.12}$$

Avec l'initialisation :

$$\beta_T(i) = 1 \text{ avec} \quad 1 \leq i \leq N \tag{2.13}$$

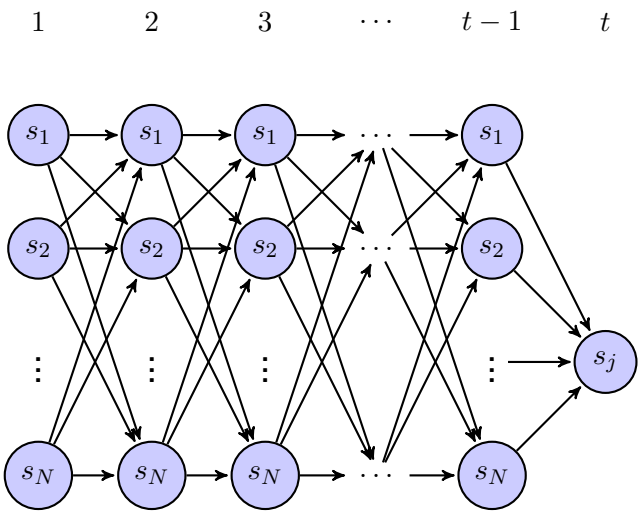


FIGURE 2.7 – Treillis illustrant le parcours effectué lors du calcul de la variable forward dans un MMC

Algorithme forward-backward L'inférence forward-backward permet de calculer la probabilité d'un état j à un instant quelconque :

$$p(X, y_t = s_j) = \alpha_t(j)\beta_t(j) = \gamma_t(j) \tag{2.14}$$

on en déduit la probabilité de la séquence observée :

$$\begin{aligned} P(X) &= \sum_j \gamma_t(j) \qquad \forall t \\ P(X) &= \sum_j \alpha_T(j) \\ P(X) &= \sum_j \beta_1(j) \end{aligned} \tag{2.15}$$

Algorithme de Viterbi L'algorithme de Viterbi a pour but de trouver la séquence d'états cachés Y qui a produit avec la plus grande probabilité la séquence observée X, selon un modèle M. C'est à dire :

$$v = \max_Y P(Y, X) \tag{2.16}$$

avec v la probabilité du chemin le plus probable. Une recherche du meilleur chemin est réalisée en explorant le treillis représentant l'ensemble des états possibles à chaque instant. La recherche du chemin est réalisée par une méthode de programmation dynamique [69]. L'algorithme de Viterbi peut également être utilisé en apprentissage à l'aide d'une méthode d'estimation itérative via un algorithme de type « Expectation Maximization »(EM) [50][8].

Algorithme de Baum-Welch L'algorithme de Baum-Welch [9] est utilisé pour l'apprentissage des paramètres des modèles M, à savoir la matrice de transition A, la matrice de probabilité d'observation B et le vecteur des probabilités d'états initial Π. Cet algorithme cherche à maximiser un critère, comme la vraisemblance, avec une approche EM. L'utilisation de l'algorithme forward-backward permet l'estimation des données manquantes au problème, ici les probabilités des états cachés, c'est la phase « Expectation ». Cette estimation se fait à l'aide de variables temporaires $\epsilon_t(i) = p(y_t = s_i, y_{t+1} = s_j|X, M)$ la probabilité d'être dans l'état i au temps t et dans l'état j au temps $t-1$:

$$\epsilon_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(x_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(x_t)\beta_{t+1}(j)} \tag{2.17}$$

Ces informations sont utilisées afin d'estimer les nouveaux paramètres du modèle $M = \left\{\overline{A}, \overline{B}, \overline{\Pi}\right\}$, c'est la phase « Maximization » :

$$\overline{\pi_i} = \gamma_1(i) \text{ avec } 1 \leq i \leq N \tag{2.18}$$

$$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1}\epsilon_t(i,j)}{\sum_{t=1}^{T-1}\gamma_t(i)} \text{ avec } 1 \leq i \leq N \text{ et } 1 \leq j \leq N \tag{2.19}$$

$$\overline{b_j} = \frac{\sum_{t=1,x_t=v_m}^{T-1}\gamma_t(i)}{\sum_{t=1}^{T-1}\gamma_t(i)} \tag{2.20}$$

Ce processus est répété jusqu'à atteindre un critère d'arrêt, souvent construit sur la maximisation de la vraisemblance de l'ensemble d'apprentissage.

Dans le cas d'un modèle MMC gauche droite, le modèle représente dans ce cas une séquence d'états (par exemple un état représente un caractère), l'algorithme de Baum-Welch permet alors d'effectuer une segmentation lors de l'apprentissage. En effet l'algorithme EM estime les données manquantes lors de la phase Expectation. L'algorithme de Viterbi permet aussi de produire un étiquetage des données mais en ne retenant que l'étiquetage sur le meilleur chemin il ne fournit qu'un étiquetage binaire. L'algorithme d'inférence forward-backward permet quant à lui de fournir un étiquetage probabilisé des données.

2.2.1.3 Modèles hybrides

Après avoir décrit les MMC, nous introduisons une variante des MMC, les modèles hybrides neuro-markoviens. D'une part les réseaux de neurones non-récurrents peuvent être utilisés pour classifier des vecteurs d'observations de

Modèle Gauche droite

Séquence d'états possibles à l'issue d'un alignement Viterbi sur les observations

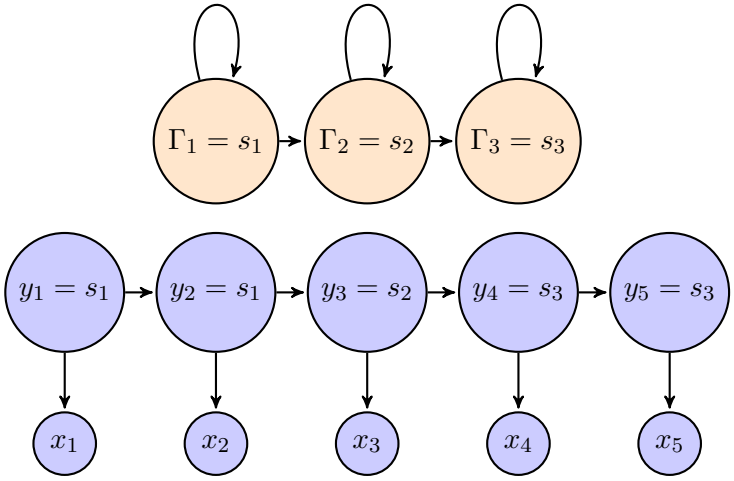


FIGURE 2.8 – Exemple d'alignement forcé entre une sortie désirée Γ de longueur 3 et une séquence de sortie Y de longueur 5

la même manière qu'un MMC, à la différence majeure que chaque décision est indépendante des autres (pas de prise en compte du contexte). Cela mène généralement à avoir des décisions qui peuvent être contradictoires lors de la classification de séquences. D'autre part, les MMC sont des modèles capables de traiter des séquences en prenant en compte le contexte, mais ils intègrent un aspect génératif de par l'utilisation des mélanges de Gaussiennes pour la modélisation des probabilités d'émissions.

Une solution est de remplacer les mélanges de Gaussiennes par un classifieur discriminant (réseau de neurones ou SVM avec des classes). Cette combinaison permet d'obtenir un étage de discrimination des vecteurs d'observations et un étage de décision au niveau de la séquence qui prend en compte les décisions successives du réseau de neurones en faisant le lien temporel (Markovien) entre ces décisions.

L'apprentissage d'un modèle neuro-markovien procède en deux temps. Cette division est réalisée puisqu'il est nécessaire pour un réseau de neurones de connaître pour chaque vecteur de données x_t le vecteur de sortie y_t associé (la classe désirée), c'est à dire associer un vecteur de données à une classe. Hors cet étiquetage des données est extrêmement fastidieux à réaliser puisqu'il faut étiqueter chaque observation à chaque instant avec la bonne étiquette. Une possibilité est d'apprendre un MMC avec des mélanges de Gaussiennes qui va réaliser un premier décodage à l'aide d'un alignement forcé de la base d'apprentissage afin de réaliser l'association mentionnée. Le modèle neuro-markovien peut ensuite être appris de manière itérative, selon l'algorithme EM, en alter-

nant une phase d'apprentissage et de génération des probabilités a posteriori par le réseau de neurones et une phase d'apprentissage et de décodage par le MMC (Viterbi, ou forward-backward). L'algorithme 1 décrit l'apprentissage de ce modèle [14].

Apprendre un modèle MMC seul (pour le premier décodage).
Tant que (Non stagnation de la vraisemblance) **faire**
    Décoder la base d'apprentissage à l'aide du modèle (avec l'inférence Viterbi).
    Apprendre le réseau de neurones sur le décodage existant de la base.
    Obtenir les probabilités caractères du réseau de neurones sur la base.
    Apprendre le MMC en utilisant les probabilités caractères en tant que vraisemblance.
    Mesurer le taux d'erreur sur une base de test.
Fait
.

Algorithme 1 – Apprentissage d'un modèle hybride neuro-markovien

Une variante est de procéder selon une inférence forward-backward au lieu de Viterbi. On obtient dans ce cas un étiquetage probabilisé des observations, et c'est $\gamma_t(j)$ qui est utilisé comme fonction objectif pour l'apprentissage du réseau de neurones [149].

Bien que cette méthode de classification dynamique corrige l'aspect génératif du MMC elle reste fondée sur des décisions locales prises indépendamment les unes des autres par le réseau de neurones. Il existe d'autres modèles hybrides, tels que des combinaisons SVM-MMC ou CAC-MMC, ils fonctionnent d'une manière très similaire aux modèles neuro-markoviens et avec les mêmes désavantages.

Les bases algorithmiques sur les processus stochastiques de génération des observations par des états sont aussi utilisées dans le cadre des modèles de Champs Aléatoires Conditionnels que nous décrivons maintenant.

2.2.2 Champs Aléatoires Conditionnels

Les CAC [111] sont des modèles dynamiques discriminants. Nous présentons dans un premier temps le modèle du CAC puis nous décrivons les méthodes d'apprentissage de celui-ci.

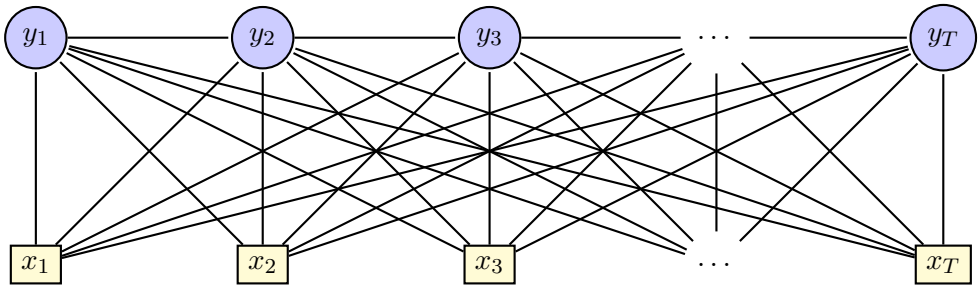


FIGURE 2.9 – Lien entre les observations x_t et les états y_t dans un CAC

2.2.2.1 Modèle

Comme les MMC il s'agit de modèles probabilistes, adaptés aux données séquentielles, avec des observations X et des états cachés Y. La différence majeure est qu'un CAC modélise la probabilité d'avoir une séquence d'états cachés, sachant la séquence des observations, c'est à dire la probabilité $p(Y|X)$, au lieu de $p(Y, X)$ pour un MMC. Il s'agit donc d'un modèle discriminant et non génératif, contrairement aux MMC. Il génère une frontière de décision entre les différentes classes, comme un réseau de neurones et modélise les transitions entre les états.

Comme le MMC, le CAC a une dépendance markovienne à l'ordre un entre les états cachés. Il est important de noter que la séquence d'observations X a une influence sur l'ensemble de la séquence de label Y, le CAC exploite l'ensemble des informations de la séquence X pour effectuer une décision locale.

Le modèle graphique illustrant les CAC est présenté en figure 2.9, le lien entre les observations et les états est non orienté par rapport à la figure 2.6 puisque le CAC est un modèle non orienté, chaque arc est pondéré d'une fonction de potentiel réel.

L'équation 2.21 présente le modèle CAC [111, 125].

$$p(Y|X) = \frac{1}{Z(X)} \exp\left(\sum_{t=1}^{T}\sum_{i=1}^{I} \lambda_i f_i(y_{t-1}, y_t, X, t)\right)) \tag{2.21}$$

$$Z(X) = \sum_{y'} \exp\left(\sum_{t=1}^{T}\sum_{i=1}^{I} \lambda_i f_i(y'_{t-1}, y'_t, X, t)\right) \tag{2.22}$$

$Z(x)$ est un terme de normalisation sur l'ensemble des états possibles sur le treillis Y, les $f_i(y_{t-1}, y_t, X, t)$ sont des fonctions de caractéristiques et les λ_k sont les poids des fonctions de caractéristiques. Parmi les fonctions de caractéristiques il faut distinguer d'un côté les fonctions de transitions $f_i(y_{t-1}, y_t)$ (la dépendance markovienne d'ordre 1) et de l'autre les fonctions d'observation

$f_i(y_t, X)$. Les fonctions de transitions sont des fonctions binaires. Les fonctions d'observations peuvent être des fonctions réelles, discrètes ou binaires. Comme on peut le voir sur la figure 2.9, les CAC ne font pas l'hypothèse d'indépendance des observations conditionnellement aux états. Ils ont donc la capacité d'utiliser un très large contexte dans les données d'entrée pour prendre une décision locale.

Contrairement aux MMC (présentés en section 2.2.1) ce sont des modèles discriminants, évitant donc de modéliser des probabilités a priori $p(X)$ et $p(Y)$. C'est un avantage important pour un problème de classification, car bien qu'il soit possible d'utiliser des modèles génératifs pour effectuer de la classification de séquences, les modèles discriminants modélisent la vraie fonction de décision et ils sont optimisés dans cet objectif. Les CAC semblent donc plus intéressants que les MMC pour la classification de séquences, cependant ils possèdent deux inconvénients majeurs :

— ils ne réalisent pas l'étiquetage au cours de l'apprentissage ;
— ils ne possèdent pas d'états interne leur permettant de structurer les observations en données de plus haut niveau.

Deux variantes des CAC existent pour corriger ces problèmes. Les CAC à états cachés [143], rajoutent une couche d'états cachés aux CAC qui fonctionne de manière analogue à celle d'un MMC. Cette couche cachée permet de modéliser les observations, et donc d'apprendre à structurer les informations. Les CAC latent-dynamiques [133] permettent quant à eux de réaliser un alignement séquence–label durant l'apprentissage.

Nous présentons maintenant plusieurs méthodes d'apprentissages des CAC.

2.2.2.2 Apprentissage

L'optimisation d'un CAC pour une tâche de classification passe par une optimisation des λ_k pour maximiser un critère sur une base d'apprentissage $\mathcal{B}$:

$$\mathcal{B} = (\{X_1, Y_1\}, \{X_2, Y_2\}, \dots \{X_p, Y_p\} \dots \{X_P, Y_P\}) \tag{2.23}$$

avec P le nombre d'éléments dans la base. Le critère le plus utilisé est la log-vraisemblance :

$$l = \sum_{n=1}^{N} \log p(Y_p|X_p) = \sum_{p=1}^{P} \left(\sum_{t=1}^{T} \sum_{i=1}^{i} \lambda_i f_i(y_{t-1,p}, y_{t,p}, X_p, t) - \log Z(X_p) \right) \tag{2.24}$$

afin de maximiser la log vraisemblance, il faut annuler sa dérivée, c'est à dire déterminer les λ_k solutions de $\nabla l = 0$. Il est donc nécessaire de calculer les dérivées partielles $\frac{\partial l}{\partial \lambda_k}$.

L'optimisation des λ_k est généralement un problème de très grandes dimensions. Il est donc nécessaire d'utiliser des algorithmes adaptés pour l'optimisation d'un très grand nombre de paramètres.

L'algorithme L-BFGS [165, 186, 196] est un algorithme itératif quasi-Newtonien permettant d'optimiser les λ_k. Il s'agit d'un algorithme de descente de gradient du second ordre, où le gradient et le pas de l'erreur sont estimés à l'aide des dérivées du second ordre. Le calcul du Hessien (la matrice des dérivées du second ordre) est trop coûteux en grandes dimensions, il est donc nécessaire d'en obtenir une estimation en estimant la courbure des gradients. Cette estimation est produite en gardant en mémoire les gradients des itérations précédentes.

La Descente de Gradient Stochastique [185](DGS), est un autre algorithme de descente de gradients, permettant d'optimiser les λ_k. La mise à jour des paramètres se fait après avoir calculé les gradients sur un sous ensemble de la base, et non pas sur chaque élément de la base. Cette mise jour moins régulière permet d'accélérer l'apprentissage, néanmoins la DGS ne permet qu'une approximation de la solution.

Les CAC sont des classifieurs dynamiques utilisant le contexte des observations pour effectuer une décision. D'autres classifieurs utilisent une mémoire interne qui résume les informations passées, c'est le cas des réseaux de neurones récurrents que nous présentons maintenant.

2.2.3 Réseaux de neurones récurrents

Les réseaux de neurones récurrents sont des classifieurs dynamiques intégrant une mémoire interne dans une couche cachée. Dans un premier temps nous présentons la notion de récurrence, ensuite nous présentons les méthodes utilisées pour apprendre de tels réseaux de neurones, puis nous présentons quatre améliorations apportées aux réseaux de neurones récurrents classiques.

2.2.3.1 Récurrence

Les réseaux de neurones récurrents sont des réseaux de neurones intégrant une mémoire [191]. Un neurone isolé est dit récurrent lorsqu'il utilise en entrée à l'instant t sa sortie à $t-1$, comme illustré sur la figure 2.10.

À l'intérieur d'un réseau récurrent au moins une couche est récurrente. C'est

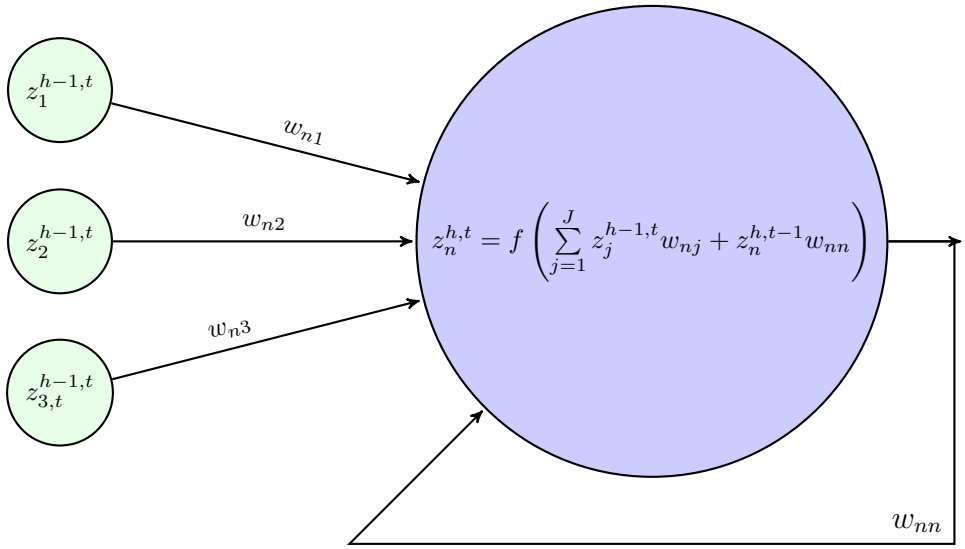


FIGURE 2.10 – Neurone récurrent

à dire que pour une couche h, les neurones n reçoivent en entrée les sorties des neurones de la couche $h-1$ à l'instant t et les sorties de tous les neurones de la couche h à $t-1$, ces connexions récurrentes sont pondérées de la même manière que les connexions non récurrentes. La somme pondérée de chaque neurone n est donc modifiée par rapport à un perceptron multi-couches, il est nécessaire d'ajouter un terme afin de prendre en compte la récurrence :

$$z_n^{h,t} = \sum_{j=1}^{J} (z_j^{h-1,t} w_{nj}) + \sum_{r=1}^{R} (z_r^{h,t-1} w_{nr}) \tag{2.25}$$

avec J le nombre de neurones sur la couche $h-1$ et R le nombre de neurones sur la couche h.

Cette prise en compte du contexte temporel permet aux réseaux de neurones une meilleure performance sur des problèmes pour lesquels l'aspect temporel est important, comme par exemple la reconnaissance de l'écriture.

2.2.3.2 Apprentissage

Tout comme un réseau de neurones, l'optimisation d'un réseaux de neurones récurrents se fait pas la modification des poids de connexions afin de les optimiser sur une base d'exemples $\mathcal{B}$. Il existe deux façons principales d'optimiser un réseau récurrent : la rétro-propagation temporelle [189](« Back Propagation Throuh Time », BPTT) ou la rétro-propagation en temps réel [190](« Real Time Back Propagation », RTBP). La BPTT est très similaire à un apprentissage d'un réseau de neurones profond via une rétro-propagation sur la totalité

du réseau. Étant donné un nombre de pas temporels P, le réseau est « déplié », voir la figure 2.11. Ainsi de $t - P$ à t les couches récurrentes deviennent des couches non-temporelles connectées à la sortie et à une instance future d'elle même. Les connexions récurrentes sont donc dupliquées plusieurs fois, elles sont illustrées en orange sur la figure 2.11. L'apprentissage des poids se fait alors par une rétro-propagation standard, où les poids d'une couche récurrente sont moyennés à chaque itération d'apprentissage.

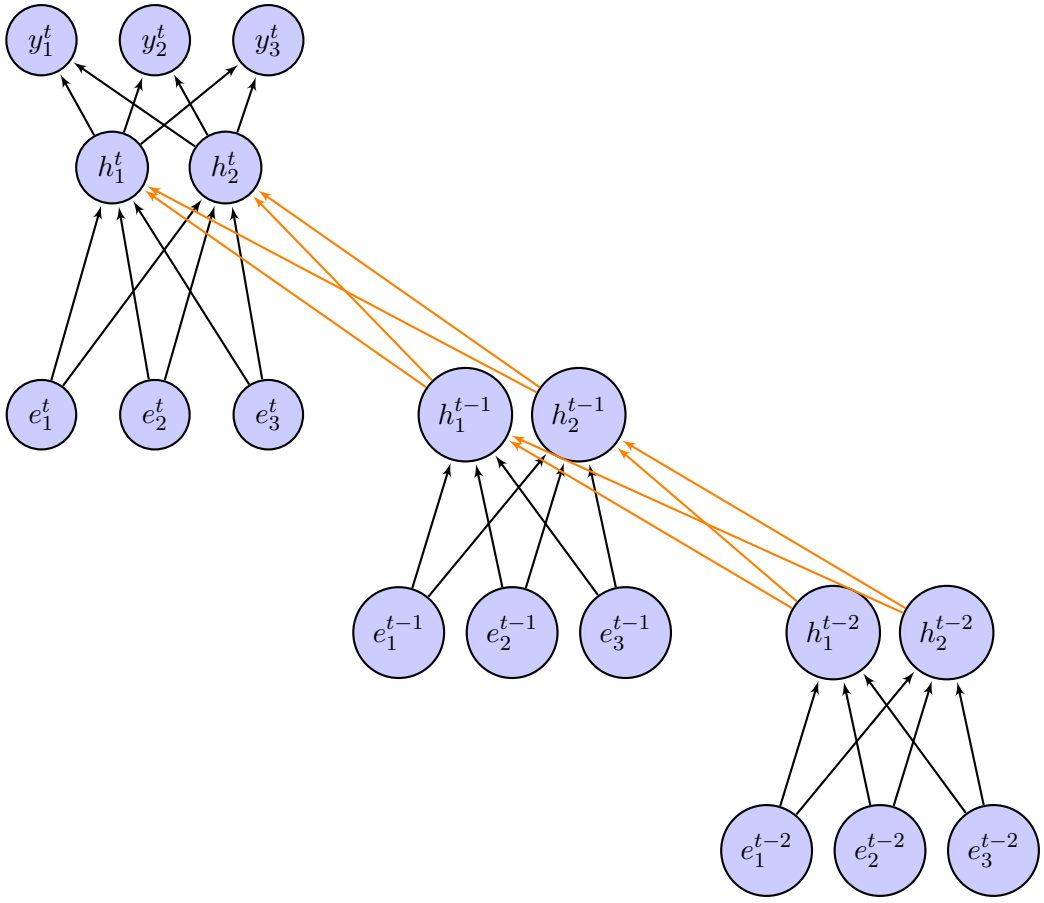


FIGURE 2.11 – Exemple de dépliage du BPTT avec un pas de 2, les connexions récurrentes dupliquées au moment du dépliage sont représentées en orange.

La RTRL[193] est une approche plus mathématique du problème, là où le BPTT est une approche très graphique. L'objectif est de minimiser l'erreur quadratique instantanée de sortie [123], les pondérations sont donc adaptées à chaque instant. Néanmoins quel que soit l'apprentissage utilisé, il est constaté que l'une des faiblesses des réseaux récurrents, utilisant des fonctions de transferts classiques, est la disparition du gradient [71, 83]. Il s'agit du même phénomène constaté sur les réseaux profonds. Sans un algorithme adapté ou une gestion différente de la mémoire, le gradient de l'erreur diminue vite durant la

rétro-propagation de l'erreur. Ce problème a un impact au moment de l'utilisation du réseau récurrent car les poids favorisent le contexte récent et non le contexte lointain (plus d'une dizaine d'unités temporelles). Nous décrivons maintenant une unité neuronale beaucoup moins sensible à la disparition du gradient de par sa construction.

2.2.3.3 Neurones Long Short Term Memory

Une solution proposée pour réduire la perte du gradient et modéliser des dépendances plus longues, est l'utilisation de neurones « Long Short Term Memory » (LSTM) [77, 94]. Ces neurones sont plus complexes qu'une simple fonction de transfert, comme peut l'illustrer la figure 2.12. Ils sont composés d'une unité de mémoire et de quatre portes, dont trois portes de contrôles, et de trois opérateurs. Une porte de contrôle est un neurone récurrent avec une fonction de transfert bornée $[0, 1]$, connecté aux entrées de la couche ainsi qu'à l'ensemble des sorties précédentes produites par les neurones de la même couche et à la mémoire des unités de cette couche. La propagation en avant du signal au travers d'un neurone LSTM se fait de la manière suivante :

1. calcul de l'entrée :

$$p_n^{h,t} = f_\mu \left(\sum_{j=1}^{J} (z_j^{h-1,t} w_{nj}) + \sum_{r=1}^{R} (z_r^{h,t-1} w_{nr}) \right) \tag{2.26}$$

   avec J le nombre de neurones de la couche $h-1$, et R le nombre de neurones sur la couche h de LSTM ;

2. calcul de la porte de contrôle des entrées :

$$g_{\gamma n}^{h,t} = f_\gamma \left(\sum_{j=1}^{J} (z_j^{h-1,t} w_{\gamma j}) + \sum_{r=1}^{R} (z_r^{h,t-1} w_{\gamma r}) + \sum_{l=1}^{L} (c_l^{h,t-1} w_{\gamma l}) \right) \tag{2.27}$$

   avec L le nombre de neurones sur la couche ;

3. calcul de la porte doubli :

$$g_{\phi n}^{h,t} = f_\phi \left(\sum_{j=1}^{J} (z_j^{h-1,t} w_{\phi j}) + \sum_{r=1}^{R} (z_r^{h,t-1} w_{\phi r}) + \sum_{l=1}^{L} (c_l^{h,t-1} w_{\phi l}) \right) \tag{2.28}$$

4. calcul de l'état de la mémoire :

$$c_n^{h,t} = g_{\phi n}^{h,t} c_r^{h,t-1} + g_{\gamma n}^{h,t} p_n^{h,t} \tag{2.29}$$

5. calcul de la porte de contrôle de la sortie :

$$g_{\omega n}^{h,t} = f_\omega \left(\sum_{j=1}^{J} (z_j^{h-1,t} w_{\omega j}) + \sum_{r=1}^{R} (z_r^{h,t-1} w_{\omega r}) + \sum_{l=1}^{L} (c_l^{h,t-1} w_{\omega l}) \right) \quad (2.30)$$

6. calcul de la sortie de la cellule :

$$z_n^{h,t} = g_{\omega n}^{h,t} f_c(c_n^{h,t}) \quad (2.31)$$

avec f_c une fonction de transfert appliquée sur la valeur de la cellule, dans le but de borner la valeur de la cellule.

La porte d'entrée contrôle l'influence du signal entrant sur l'état de la mémoire : si un signal n'est pas considéré comme pertinent la porte d'entrée va produire une sortie proche de 0, empêchant le signal $p_n^{h,t}$ d'être sommé à la mémoire. La porte d'oubli contrôle l'influence de l'état précédent de la mémoire sur le nouvel état, elle peut effacer la mémoire ou la conserver. La porte de sortie contrôle l'influence de ce neurone sur le reste du réseau, cela permet de stocker des informations pertinentes pour un instant $t' \neq t$ sans avoir à présenter l'état de la mémoire au reste du réseau.

Une couche de neurones LSTM possède donc quatre fois plus de connexions en entrée qu'une couche classique. Malgré cet accroissement des connexions un réseau avec une couche LSTM peut toujours être entraîné par BPTT. L'ensemble des portes permet un contrôle très fort sur le contenu de la mémoire et réduit de manière très importante la perte de gradient (il n'y a pas de preuves démontrant ce phénomène, seul un constat empirique en a été fait). Cette amélioration importante permet aux réseaux de neurones récurrents d'exploiter des informations sur de très grandes périodes temporelles. Ces améliorations ont été intégrées dans des architectures plus complexes de réseaux récurrents que nous présentons maintenant.

2.2.3.4 Bidirectional LSTM

Les Bidirectional LSTM [77] (BLSTM) sont une amélioration supplémentaire des réseaux récurrents avec des neurones LSTM. Il s'agit, dans le cas d'applications temporelle, de combiner un réseau récurrent composé de neurones LSTM exploitant les données dans le sens chronologique avec un second réseau récurrence composé de neurones LSTM exploitant les données dans le sens antéchronologique. Cela permet lors d'une prise de décision à l'instant t de disposer du contexte passé et du contexte futur. Dans le cas de la recon-

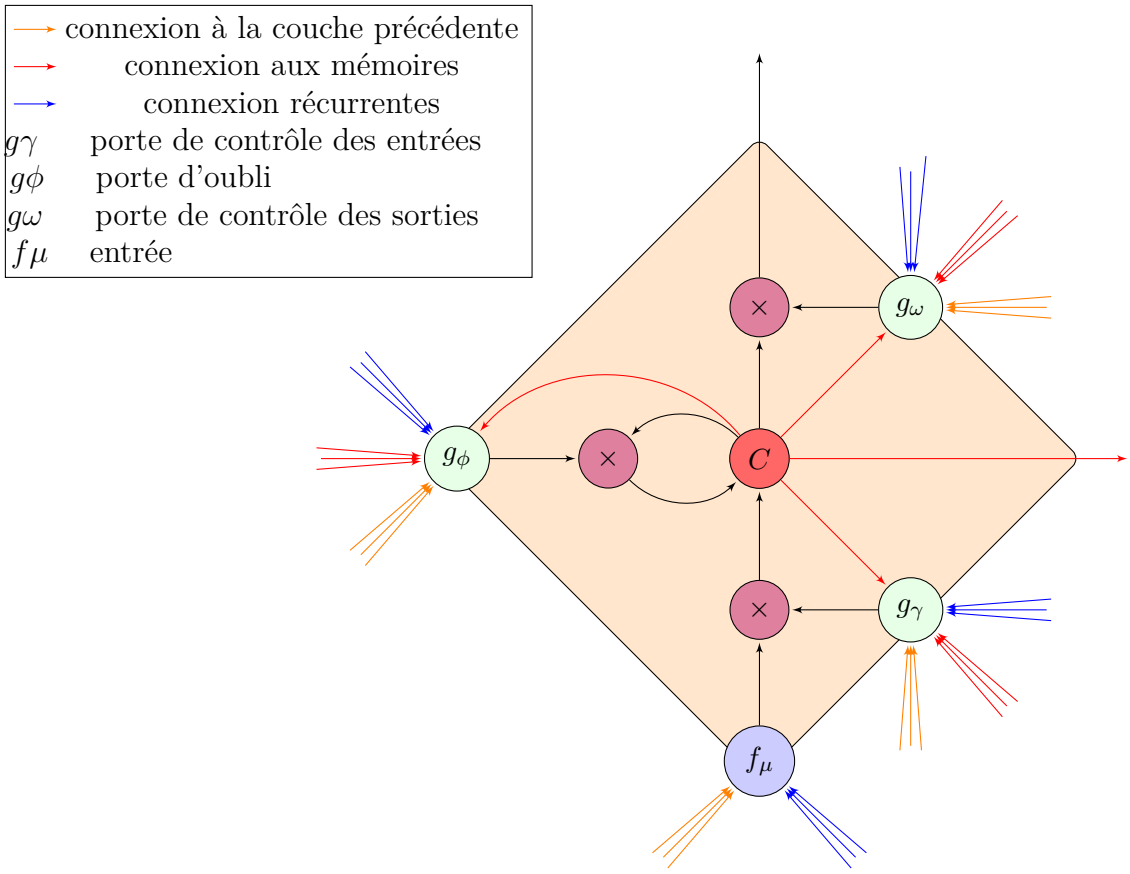


FIGURE 2.12 – Un neurone LSTM

naissance de l'écriture il s'agit d'une information intéressante car elle permet de prendre une décision sur un caractère avec une vision des caractères voisins, à gauche et à droite.

Une amélioration supplémentaire de ce système est l'extension du BLSTM à des signaux multi-dimensionnels. Nous la présentons maintenant.

2.2.3.5 MDRNN

Le classifieur suivant est issu des travaux présentés dans [84]. Il s'agit du classifieur qui fait état de l'art. Ce réseau récurrent est une extension du modèle BLSTM à un signal 2D, le modèle BLSTM ne parcourant qu'une seule dimension.

L'intelligence de ce classifieur est concentré dans le modèle « Multi-dimensional Recurrent Neural Network » (MDRNN). L'idée la plus importante de ce classifieur est d'associer à chaque dimension du problème deux réseaux de neurones récurrents. L'un des réseaux parcourt la dimension dans une direction, et le se-

cond dans l'autre direction. Ainsi dans une image il lie la dimension horizontale et la dimension verticale.

Un second concept important de cette approche est la hiérarchisation du réseau. Cette hiérarchisation permet d'abstraire l'information à différents niveaux. Le réseau est bâti suivant un modèle de réseau de neurones convolutionnelle :

1. l'image est subdivisée en sous bloc de $N \times M$, ces sous blocs sont transformés en vecteurs ;
2. quatre réseaux LSTM (MDLSTM) parcourent l'image dans chaque direction (du coin supérieur gauche vers le coin inférieur droit, du coin inférieur gauche vers le coin supérieur, etc .) pour produire une abstraction de chaque sous-bloc de l'image ;
3. les sorties de plusieurs LSTM parcourant l'image dans le même sens sont assemblées et passées au travers d'une couche convolutionnel ;
4. une étape de sous échantillonage génère une image de dimension inférieure à la première mais contenant des informations de plus haut niveau.

Dans cette hiérarchisation la sortie de chaque réseau devient une caractéristique pour le réseau suivant. Finalement les sorties des derniers réseaux sont propagés au travers une couche non récurrente de lissage spatiale et de sous échantillonnage pour produire une représentation plus abstraite des informations.

La figure 2.13 provenant de [118] illustre cet enchaînement de couches.

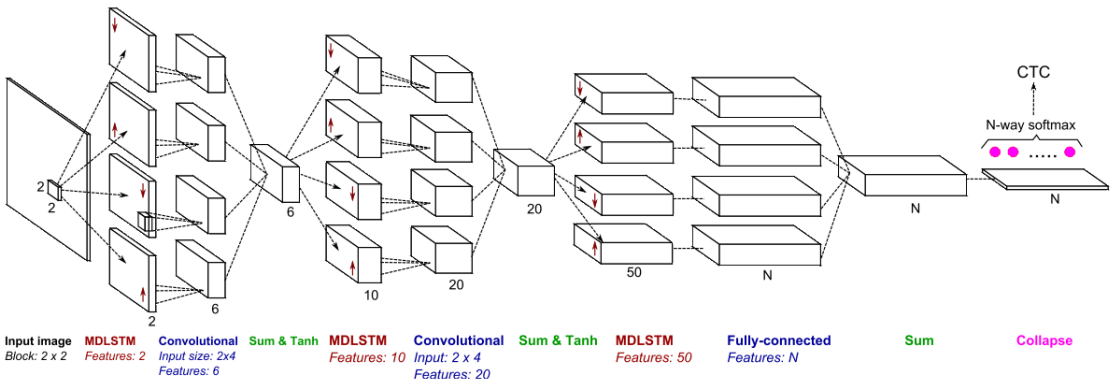


FIGURE 2.13 – Un réseau MDRNN-CTC, image provenant de [118]

Ce processus est répété autant de fois que nécessaire pour obtenir un niveau d'abstraction et une résolution des images assez faible pour résoudre le

problème étudié. Le réseau est généralement construit suivant une pyramide inversée, les blocs LSTM avec un petit nombre de neurones sont situés sur les couches les plus basses du réseau et sont chargés de transformer les informations de bas niveau en information de moyen niveau, tandis que les blocs LSTM avec le plus grand nombre de neurones sont situés sur les couches les plus hautes et sont chargés de transformer les informations de moyen niveau, provenant de la couche précédente, en information de haut niveau. Ces informations sont propagées à la couche de décision qui est mono-dimensionnelle. Ce qui permet d'obtenir un réseau dont la majorité des poids, et donc du traitement de l'information se fait sur la couche haute. Les couches basses réalisent une abstraction des caractéristiques de plus en plus importante, de manière très similaire à un réseau de neurones profond.

Les réseaux BLSTM, MDRNN et de manière générale les réseaux récurrents, n'ont pas d'algorithmes dédiés permettant de segmenter une sortie désirée Γ d'une longueur $G = |\Gamma| \leq T$ et $\Gamma_k \in \{s_1, s_2, \ldots, s_K, \}$ afin de la transformer en une séquence de sortie Y avec $|Y| = |X| = T$, tout comme le CAC il est donc nécessaire de générer cette segmentation. Une adaptation de l'algorithme forward-backward des MMC a été réalisée afin d'intégrer durant l'apprentissage une segmentation permettant à tout instant t d'obtenir une erreur de sortie. Nous présentons maintenant cette amélioration.

2.2.3.6 Couche de classification Connexionniste

La couche de classification connexionniste (« Connectionist Temporal Classification », CTC) est une couche de réseau de neurones dédiée à la classification. Le CTC [80] est une couche de réseau de neurones de type « SoftMax », la fonction de transfert (équation 2.32) de cette couche normalise les sorties dans l'intervalle $[0, 1]$ et la somme des sorties est égale à 1.

$$z_k^t = \frac{\exp^{a_k^t}}{\sum_{}^{K} \exp^{a_k^t}} \tag{2.32}$$

Avec z_k^t la sortie du neurone k au temps t et a_k^t la somme pondérée de l'entrée du neurone k au temps t. Cette couche comporte $|K| + 1$ neurones. Chaque neurone représente une classe du problème et le dernier est un symbole joker, représentant une absence de décision ou un blanc. Cela permet par exemple d'éviter de prendre des mauvaises décisions sur des cas complexes.

Il n'y a aucune intégration du contexte dans une couche CTC, il n'y a ni récurrence ni unité de mémoire. Sa capacité à traiter les séquences vient des

deux propriétés suivantes :

1. le réseau de neurones récurrent sous–jacent ;
2. la fonction objectif, qui joue un rôle important lors de la rétro-propagation des erreurs, au moment de l'apprentissage.

La définition d'une fonction objectif capable de définir à chaque temps une cible « réaliste » par rapport à la séquence observée est donc importante. L'erreur label LER est mesurée en fonction d'une distance d'édition ED, comme par exemple la distance de Levenshtein [160]. La fonction $h : X \to \Gamma$ du CTC est paramétrée de manière à minimiser l'erreur label LER étant donné une base de test $S \subset D_{X \times \gamma}$:

$$LER(h, S) = \frac{1}{|S|} \sum_{(X,\gamma) \in S} \frac{ED(h(X), \gamma)}{|\gamma|} \tag{2.33}$$

La fonction objectif a pour but de minimiser cette erreur. On considère un mot γ, de longueur l sur les classes C, qui représente la séquence de sortie cible, avec en signal d'entrée la séquence X de longueur T, qui représente la séquence des trames observées. Afin de rétro-propager l'erreur il est nécessaire de définir à chaque temps une cible o_k^t pour la sortie y_k^t de chaque neurone du CTC. Les sorties au cours temps du CTC sont définies comme le produit des probabilités conditionnelles sur un chemin π, selon equation 2.34.

$$p(\pi|x) = \prod_{t=1}^{T} p(\pi_t, t|X) = \prod_{t=1}^{T} y_{\pi_t}^t \tag{2.34}$$

Un opérateur B de simplification des chemins est utilisé : il supprime les absences de décisions et ne conserve qu'une seule instance de lettre d'une séquence continue de ce caractère. Ainsi :

$$B(\_, a, \_, \_, \_, a, \_, \_, \_, b) = B(\_, a, a, \_, \_, a, \_, \_, b, \_) = (a, a, b) \tag{2.35}$$

Les chemins étant mutuellement exclusifs ils décrivent donc un label l qui est une séquence de caractères : $p(l|x) = \sum_{\pi \in B^{-1}(l)} p(\pi|X)$. La probabilité d'une séquence de caractères l est donc la somme des probabilités de tous les chemins qui peuvent la générer. La difficulté de situer avec précision un label de classe C dans une séquence est donc évitée, car la somme couvre l'ensemble des zones où il est possible de trouver ce label. Calculer l'ensemble des positions reste cependant une tâche trop complexe pour être accomplie de manière naïve. Une modification de l'algorithme forward-backward des MMC est proposée [82]

pour résoudre ce problème via l'utilisation de la programmation dynamique. La fonction objectif déterminée par les équations précédentes permet de rétro-propager les erreurs de la couche CTC vers les couches du réseau récurrent.

Il est intéressant de constater l'effet engendré par la fonction objectif sur les sorties, une fois le réseau entraîné [80]. Le label blanc est détecté très régulièrement alors que les classes sont détectées de manière sporadique. Ces détections sont toujours très prononcées, quasiment binaire au niveau de la sortie du CTC.

Après avoir présenté un ensemble de classifieurs dynamiques nous présentons maintenant un tableau récapitulatif mettant en avant leurs différences fonctionnelles.

2.3 Différences fonctionnelles des classifieurs

Dans le tableau 2.1 nous listons un ensemble d'avantages et d'inconvénients des différents classifieurs ou de combinaison de ces classifieurs dans le cadre de la reconnaissance de l'écriture. Nous comparons les éléments suivants :

Le type d'entrée : un classifieur peut prendre en entrée un vecteur de réels, un vecteur de valeurs discrètes, etc. ;

Dimension des entrées : certains algorithmes supportent plus ou moins bien les grandes dimensions pour les vecteurs d'entrée. Un petit vecteur d'entrée possède une dimension inférieure à 100. Un grand vecteur possède une dimension comprise entre 100 et 2000 tandis qu'un très grand vecteur possède une dimension supérieure à 2000 ;

Structuration spatiale de l'information : le classifieur possède une capacité à encoder l'information spatiale dans le système ;

Structuration temporelle de l'information : le classifieur possède une capacité à utiliser le contexte ou possède une mémoire interne ;

Discrimination locale forte : le classifieur émet une probabilité a posteriori pour effectuer la décision d'un caractère ou morceau de caractère à chaque instant.

Segmentation du signal désiré : si le classifieur possède un algorithme capable de générer une segmentation lors de l'apprentissage (un étiquetage des données) ;

Ce tableau ne permet pas de déterminer si un classifieur est plus performant qu'un autre, il ne rend compte que des différences à certains points de

Classifieur	Type d'entrée	Dimension du vecteur d'entrée	Classifieur dynamique	Structuration spatiale de l'information	Discrimination locale forte	Segmentation du signal désiré
CAC	$S = \{1 \dots N\}$	Très grande	Oui	Non	Oui	Non
MMC	$\mathbb{R}$	Petite	Oui	Non	Non	Oui
MMC - Réseau de neurones	$\mathbb{R}$	Grande	Oui	Moyenne	Non	Oui
Réseau de neurones	$\mathbb{R}$	Grande	Non	Moyenne	Non	Non
Réseau de neurones profond	$\mathbb{R}$	Très grande	Non	Forte	Oui	Non
Réseau de neurones récurrent	$\mathbb{R}$	Grande	Oui	Moyenne à forte dépendant du nombre de couches	Oui	Oui
BLSTM	$\mathbb{R}$	Grande	Oui	Oui	Oui	Non
BLSTM - CTC	$\mathbb{R}$	Grande	Oui	Oui	Oui	Oui
MDRNN	$\mathbb{R}$	Grande	Oui	Forte	Oui	Non
MDRNN - CTC	$\mathbb{R}$	Grande	Oui	Forte	Oui	Oui

TABLE 2.1 – Récapitulatifs des avantages et inconvénients des classifieurs

fonctionnement. Il permet donc d'aider à choisir un classifieur étant donné un certain nombre de contraintes du système. Nous nous appuierons sur ce tableau pour justifier des choix de classifieurs dans le domaine de la reconnaissance de l'écriture manuscrite.

2.4 Conclusion

Nous avons présenté un ensemble de méthodes de classifications statistiques différentes. Chaque méthode possède des avantages et des inconvénients, rappelés en partie dans le tableau 2.1, qui destinent l'utilisation de cette méthode à différentes applications. Nous avons en particulier insisté sur les différences entre les classifieurs statiques, qui n'ont aucune mémoire ou contexte temporel, et les classifieurs dynamiques, qui ont une mémoire ou utilisent le contexte temporel. Les classifieurs dynamiques nous sont d'un intérêt particulier puisque la reconnaissance de l'écriture manuscrite est une application nécessitant la prise en compte du contexte temporel au niveau des entrées (relation d'un vecteur de données à un autre) et au niveau des sorties (relation d'un caractère à un autre).

Dans le chapitre suivant nous décrivons plus en détail les systèmes de reconnaissance de l'écriture. Nous présentons l'historique du domaine ainsi que les algorithmes et systèmes utilisés.

Chapitre 3

Systèmes de reconnaissance de l'écriture manuscrite

Le chapitre précédent présente un ensemble de méthodes de classification statistiques permettant d'associer un vecteur d'observations à une séquence de sorties représentant des classes. Nous avons présenté des méthodes de classification statistiques permettant d'aborder des problèmes statiques, c'est à dire sans lien temporel entre les éléments d'une séquence, ainsi que des méthodes de classification dynamiques permettant de traiter une séquence dont les éléments ont un lien temporel. Dans ce chapitre nous présentons des systèmes permettant d'effectuer la reconnaissance de l'écriture, ces systèmes s'appuient sur les méthodes de classification présentées précédemment afin de transformer une séquence de vecteur de données (appelé séquence de vecteur de caractéristique dans le domaine de la reconnaissance d'images) extrait d'une image de mot ou de phrase, en une séquence de caractères désignant ce mot ou cette phrase.

La reconnaissance automatique de l'écriture manuscrite est le processus permettant de passer d'une image, un signal $2D$, contenant une information textuelle à un texte en format électronique. Transformer un document papier en un document numérique est intéressant dans de nombreux domaines. En effet un document numérique possède l'avantage de pouvoir être traité par des systèmes d'information pour assister un opérateur humain. Les documents numériques sont donc de plus en plus communs, mais dans de très nombreux cas l'utilisation d'un document manuscrit est inévitable : adresses postales, chèques bancaires, formulaires cerfa, etc. . Transformer ces documents papier en documents numériques permet donc leur traitement par un système d'information. Le routage du courrier [60] est automatisé via des systèmes contenant un moyen automatique de reconnaissance de l'écriture, de même pour l'encaissement des chèques de faibles valeurs [104].

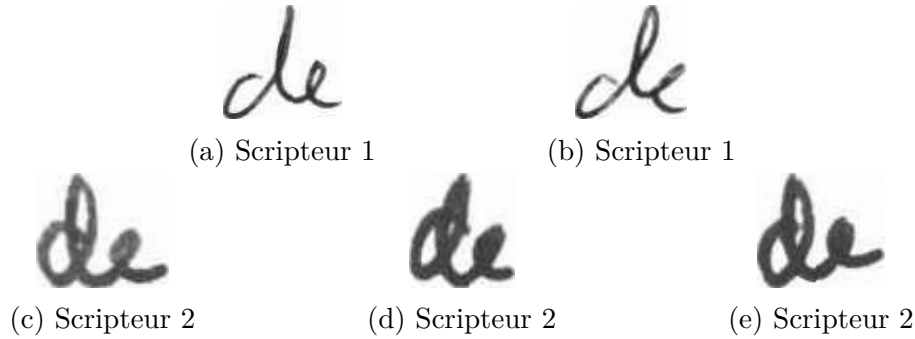
Dans ce chapitre bibliographique nous commençons par un rapide historique présentant l'ordre d'apparition des différentes techniques de reconnaissance de l'écriture. Nous examinerons par la suite l'architecture d'un système

de reconnaissance automatique, pour ensuite détailler chaque partie de ce système : prétraitements, espace de représentation, post-traitements ; nous présenterons finalement des systèmes complets de reconnaissance.

3.1 Historique

Les premiers pas dans le domaine de la reconnaissance de l'écriture manuscrite sont anciens [135]. Ils font appel à des systèmes mécaniques simples mais peu performants et lents. Les progrès dans ce domaine sont longs et complexes, de part la nature de l'écriture : la variabilité inter-scripteur et intra-caractères est très importante, comme l'illustre la figure 3.1. Le scripteur 1 et 2 n'écrivent jamais le mot « de » de manière identique, il y a de légères variations comme des boucles ou une pente différente de l'écriture. Au fil du temps, cette variabilité pousse de plus en plus à délaisser des techniques simples comme l'utilisation des masques, au profit de systèmes dits « intelligents ». Ces systèmes s'adaptent au style du scripteur, utilisent le contexte local et global pour prendre une décision, font appel à des dictionnaires et des modèles linguistiques.

Ce bref historique mêle à la fois des techniques de reconnaissance manuscrite et dactylographiée, les techniques étant identiques et généralement transposables d'un style à l'autre. La figure 3.2 présente une frise chronologique des différentes périodes, avec les publications importantes et les différentes ruptures technologiques.



(a) Scripteur 1 (b) Scripteur 1

(c) Scripteur 2 (d) Scripteur 2 (e) Scripteur 2

FIGURE 3.1 – Variabilité de l'écriture manuscrite

Nous nous basons sur cette figure pour présenter l'historique suivant les quatre grandes phases : les pionniers, les modèles statistiques, les modèles markoviens hybrides, et l'essor des réseaux de neurones.

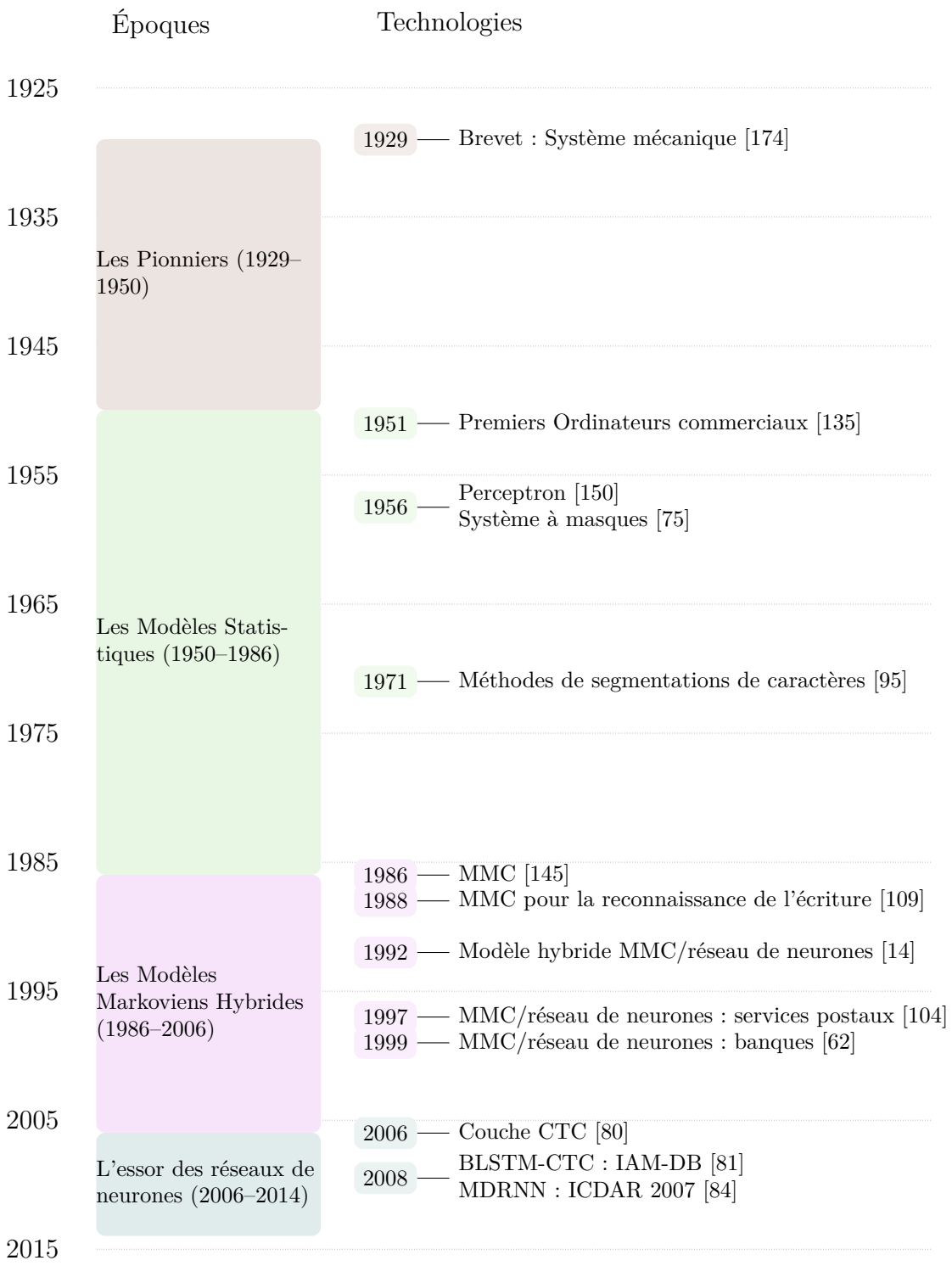


FIGURE 3.2 – Frise simplifiée de la reconnaissance de l'écriture

3.1.1 1929 à 1950 : les pionniers

Le premier brevet pour un système de reconnaissance de l'écriture est déposé en 1929 par Tauschek [174]. La solution proposée [135] consistait en un mécanisme équipé d'une source lumineuse et de masques correspondant aux différents caractères à reconnaître. La reconnaissance s'effectue en faisant passer la source lumineuse au travers des différents masques et du caractère à reconnaître. Si le résultat en sortie était une absence totale de lumière alors le masque représentait la lettre. C'est une version mécanique du filtrage par motif. Elle ne fonctionne que par un positionnement exact des lettres et pour un seul type de police de caractères. Ce système n'est adapté qu'à la reconnaissance de l'écriture dactylographiée, il ne peut pas supporter la variabilité de l'écriture manuscrite.

En 1950 avec l'arrivée des premiers ordinateurs, et la simplification des calculs complexes qu'ils apportent, la reconnaissance automatique de caractères devient un centre d'intérêt pour de nombreux chercheurs. Les premiers systèmes complets de reconnaissance sont des transferts de la technologie mécanique vers la technologie électronique. Ces systèmes incluent soit des scanners à tambours soit un système optique semblable à un masque[135]. Ils fonctionnent sur le même principe de filtrage par motif : une image de caractère est comparée à un ensemble de masques, la lettre rattachée au masque produisant le recouvrement maximal est affectée à l'image du caractère inconnu. Ce sont des systèmes traitant uniquement le dactylographié et ils ne sont capables de traiter correctement qu'un faible nombre de polices, sur des textes non dégradés (peu ou pas de bruit, pas d'inclinaison du caractère). Ils ne sont toujours pas adaptés à l'écriture manuscrite.

3.1.2 1950 à 1986 : les modèles statistiques

Des modèles statistiques sont introduits par la suite pour remplacer les masques, comme : les perceptrons [126, 150], K-plus proches voisins [188]. Ces modèles statistiques permettent de remédier au défaut d'adaptation des masques statiques, c'est à dire qu'ils supportent mieux le bruit ou la rotation, et sont plus à même de s'adapter aux variations de l'écriture. Ces méthodes sont donc applicables sur de l'écriture manuscrite. Elles nécessitent cependant une étape préalable d'extraction de caractéristiques pour construire un vecteur représentant l'image. Cette tâche va devenir un nouveau Graal des chercheurs

Cependant, qu'ils procèdent par masques ou par modèles statistiques, ces

systèmes souffrent de la nécessité de détecter le début et la fin d'un caractère pour le reconnaître. Il est donc impossible de reconnaître une séquence, c'est le paradoxe de Sayre [181] : on ne peut segmenter une séquence sans reconnaître chaque élément de cette séquence, et on ne peut reconnaître chaque élément d'une séquence sans l'avoir segmentée au préalable.

Certaines approches visent à segmenter une image en différents caractères (segmentation explicite), sans pour autant les reconnaître. La détection des points de segmentation se fait par la recherche d'informations telles que les ligatures ou les espaces inter-lettres [38, 58], à l'aide d'histogrammes de densités horizontaux [178] et d'analyses des composantes connexes [40]. L'une des premières approches de segmentation automatique [95], pour les documents imprimés, propose par la même une structure standard pour effectuer une segmentation explicite :

1. Détecter le début d'un caractère ;
2. Décider de tester une possible fin de caractère ;
3. Détecter une fin de caractère.

Bien que les résultats soient corrects pour l'écriture dactylographiée, ces approches ne sont néanmoins pas satisfaisantes pour l'écriture manuscrite. En effet, différents styles d'écritures, ainsi que certains caractères comportant des ressemblances avec des ligatures ("u","v","w","y"), rendent ces approches non viables.

3.1.3 1986 à 2006 : les Modèles Markoviens Hybrides

Motivés par les résultats des Modèles de Markov Cachés (MMC)[144, 145] dans le domaine de la reconnaissance de la parole, les MMC sont appliqués avec succès à la reconnaissance de l'écriture [109, 164]. Ils possèdent de nombreux avantages tels que l'incorporation des connaissances a priori (modèles de langages ou lexique) ou la capacité d'associer une séquence de vecteurs de caractéristiques de longueur T a une séquence de sortie de caractères de longueur L, avec généralement $T \geq L$, donc leur capacité à segmenter, c'est à dire qu'ils sont les premiers classifieurs dynamiques. Ils restent néanmoins des modèles génératifs avec une faible capacité de discrimination [7], cela est lié à leur modélisation du problème que nous expliquons en section 2.2.1.3.

Cette faiblesse peut être compensée en combinant le MMC avec des modèles discriminants [14, 28, 29, 134]. Ces nouveaux modèles hybrides permettent d'augmenter les performances. Ils peuvent être déclinés avec différents classi-

fieurs : réseaux de neurones [14, 28], séparateurs à vaste marges [2, 45, 72], réseaux profonds[176]. Cependant pour entraîner ces différents classifieurs il est nécessaire d'avoir une pré-segmentation au niveau caractère. Certaines approches proposent de réaliser des apprentissages joints [14], c'est à dire en optimisant conjointement les paramètres du MMC et du classifieur. Malheureusement ces systèmes sont tous sensibles à l'initialisation de la segmentation. Une mauvaise initialisation peut mener les paramètres du système dans un minimum local.

Néanmoins ces quelques difficultés ne sont pas des obstacles majeurs car les systèmes comportant des MMC pour la reconnaissance de caractères manuscrits ont été développés avec succès. Ces premiers systèmes hybrides sont utilisés dans des applications telles que la lecture des adresses postales [60] ou la lecture des chèques [104].

3.1.4 2006 : l'essor des réseaux récurrents

En 2008 à la suite de long travaux sur des améliorations des réseaux neuronaux [74, 80, 81, 82, 94, 158, 161], les résultats de reconnaissance progressent grandement [82, 85] via un système dit « BLSTM-CTC » dont l'apprentissage combine des algorithmes des MMC avec ceux des réseaux de neurones. De manière similaire à un MMC, ce système s'affranchit des contraintes de segmentation. Il possède en plus des frontières de décision entre caractères très fortes.

La principale force de ce système réside dans sa capacité à utiliser un contexte temporel très grand. Là où le MMC est limité à utiliser sa décision précédente, le BLSTM-CTC a la capacité de mémoriser certains éléments sur des temps bien plus grands.

L'apprentissage du modèle caractère est lié au modèle mot, ces deux modèles sont appris conjointement durant l'apprentissage. En plus d'intégrer un modèle caractère, le BLSTM-CTC peut intégrer un système de décodage dirigé par le lexique. Pour plus de précisions concernant ce système voir la section 2.2.3.

L'ensemble des améliorations effectuées dans le domaine de la reconnaissance de l'écriture permettent de constituer des systèmes complexes de reconnaissance de l'écriture. La frise 3.2 positionne ces différents systèmes dans le temps. La section suivante décrit la structure usuelle d'un système de reconnaissance de l'écriture.

3.2 Chaîne de traitements de reconnaissance de l'écriture

Les chaînes de traitement de reconnaissance de l'écriture sont généralement découpées en quatre parties. Nous allons d'abord positionner le problème et décrire certaines notations, puis nous expliquerons brièvement les quatre sous-parties de la reconnaissance de l'écriture.

3.2.1 Position du problème et notations

Le contexte de cette thèse est centré sur la reconnaissance d'un mot ou d'une phrase isolée. Nous ne décrirons pas l'ensemble des étapes en amont qui permettent par exemple de passer d'un document papier à l'extraction des couches textuelles et leur division en ligne de texte. Des informations à ce sujet peuvent être trouvées dans les publications suivantes [104, 127, 184]. Nous utilisons le terme « image » qui désigne une représentation 2D complète d'une phrase, d'un mot, ou d'un caractère tandis que le terme « imagette » désigne un sous découpage d'une image. Le terme « label » désigne un caractère c_k d'un alphabet $C = \{c_1, c_2, \ldots, c_k, \ldots, c_K\}$, comportant K classes, un « mot » est une séquence de caractères $\Gamma = (\gamma_1, \gamma_2, \ldots, \gamma_g, \ldots, \gamma_G)$ de longueur G avec $\gamma_g \in C$.

3.2.2 Chaîne de traitements

Une chaîne de traitement est divisible en quatre parties : prétraitements, extraction de caractéristiques, reconnaissance, posts-traitements.

Les prétraitements consistent en un nettoyage des données qui améliorent les étapes suivantes par la suppression d'une partie de la variabilité (bruit, inclinaison des caractères, etc.). Les images fournies en entrée d'un système de reconnaissance requièrent parfois des modifications avant de pouvoir en extraire des informations. Ces prétraitements ne sont pas toujours nécessaires, mais il a été montré qu'ils amélioraient la reconnaissance [141] en normalisant l'extraction de caractéristiques.

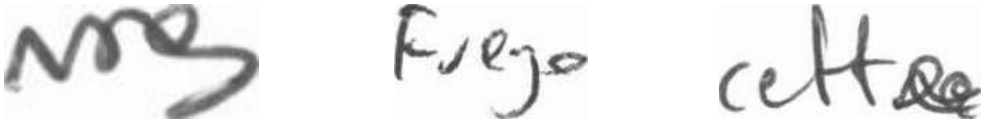
L'étape suivante consiste à extraire une séquence de vecteurs de caractéristiques décrivant l'image. Ces caractéristiques peuvent être définies par un humain, ou générées automatiquement par un algorithme.

Les caractéristiques sont ensuite fournies à un algorithme de reconnaissance, c'est à dire les méthodes statistiques de classification vues lors du cha-

pitre précédent. Les algorithmes de reconnaissance ont pour but de transformer une séquence de caractéristiques en une séquence de caractères.

La reconnaissance ne sera jamais parfaite, certaines ambiguïtés peuvent apparaître sur des mots ou des caractères, il est donc nécessaire de réaliser une étape de post-traitement. Même pour un humain, certaines images sont difficiles à identifier, c'est le cas des images en figure 3.3. Il est alors possible d'utiliser des modèles de langues pour prendre une meilleure décision ou corriger des erreurs.

Nous présentons maintenant ces différentes étapes de traitements.



(a) « mes » (b) « Fuego » (c) « cette »

FIGURE 3.3 – Cas ambigus

3.3 Prétraitements

Les prétraitements effectués sur des imagettes de mots ou de phrases ont pour but d'améliorer la reconnaissance en normalisant les images ou en réduisant la quantité de bruit dans le signal. La binarisation et la squelettisation réduisent la quantité d'information contenue dans l'image ce qui facilite la reconnaissance. La correction de pente, la correction de l'inclinaison et la normalisation ont pour but de normaliser l'aspect des images, et donc de réduire les variations inter-mots et inter-caractères. Il n'est pas nécessaire d'avoir des prétraitements, mais il a été montré que l'utilisation de prétraitements améliorent la reconnaissance [141]. La présence de certains prétraitements peut aussi être liée à l'extraction de caractéristiques. C'est le cas de la squelettisation, qui est nécessaire pour extraire des caractéristiques structurelles telles que les occlusions ou les jonctions [173].

3.3.1 Binarisation

La binarisation a pour but de transformer une image en niveau de gris en une image binaire. Cette étape est effectuée pour réduire le bruit et réduire la quantité d'information dans une image. Dépendant de la binarisation utilisée et de l'image, cette étape peut être source d'une perte d'information. Il est

important d'identifier la binarisation la plus adaptée au document traîté. La binarisation peut s'effectuer de manière simple par un seuillage statique, les valeurs supérieures à un seuil θ donné prennent une valeur binaire, les valeurs inférieures prennent l'autre valeur :

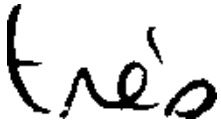
$$p'(x,y) = \begin{cases} 0 & \text{si } p(x,y) \geq \theta \\ 1 & \text{si } p(x,y) < \theta \end{cases} \tag{3.1}$$

Cette méthode est efficace pour des images très peu bruitées, cependant des ombres, ou du bruit poivre et sel peuvent persister. Voir figures 3.4b 3.4c 3.4d pour des exemples de binarisation par seuil.

Des méthodes adaptatives existent et améliorent certains résultats grâce à la prise en compte du contexte (local ou global selon les méthodes). La valeur du seuil θ n'est plus définie une fois pour toutes mais s'adapte à l'image ou à une zone de l'image. Nous pouvons par exemple citer la binarisation d'Otsu, qui prend en compte le contexte global de l'image via une étude de l'histogramme [140] pour maximiser la différence entre les deux classes binaires (figure 3.4e). La binarisation de Sauvola [157] s'adapte au niveau local en étudiant le contraste local d'une imagette de mot. La valeur seuil est soit la moyenne si le contraste est faible soit une valeur proportionnelle à l'écart type (figure 3.4f).

(a) Image originale

(b) Image seuillée à 135



(c) Image seuillée à 170

(d) Image seuillée à 235

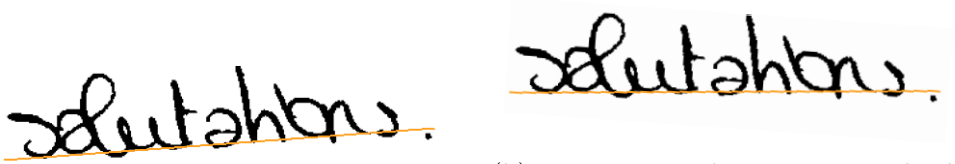
(e) Image binarisée par la méthode d'Otsu (seuil à 190)

(f) Image binarisée par la méthode de Sauvola

FIGURE 3.4 – Différentes binarisations sur une même image

3.3.2 Correction de pente

La correction de la pente modifie une image pour positionner la ligne de base à l'horizontale, la figure 3.5 illustre un exemple de correction de pente. La ligne de base est la limite inférieure du corps de l'écriture, qui est définie comme étant la zone d'une image de mot contenant la majorité des éléments des caractères. Les hampes et les jambages ne sont pas contenus dans cette zone. Détecter la ligne de base permet par la suite d'avoir des vecteurs de caractéristiques similaires pour un même caractère produit par plusieurs scripteurs. L'objectif primaire d'un algorithme de correction de pente est de déterminer l'angle de la ligne de base. Plusieurs méthodes existent pour effectuer cette tâche. Dans [136] la morphologie mathématique est utilisée pour déterminer le contour semi-convexe de l'écriture, puis la méthode des moindres carrés est utilisée pour supprimer les minima locaux. Une autre méthode consiste à étudier l'histogramme de densité horizontal d'une imagette [184], pour en déterminer le corps de l'écriture et en particulier la ligne de base. Depuis l'extraction de cette ligne de base l'ensemble des points représentants des minima locaux est déterminé. La distance moyenne entre ces points et la ligne de base est calculée de manière à supprimer les minima locaux inférieurs à cette moyenne. Cet ensemble de points permet d'obtenir par la méthode des moindres carrés une estimation de la pente. Une autre façon de déterminer les minima locaux peut consister en l'utilisation du code de chaîne des contours [122], puis l'angle est déterminé en utilisant la méthode des moindres carrés. Une méthode utilisant la transformée de Radon pour estimer la pente est proposée dans [56]. Cette méthode est plus rapide et plus robuste que les précédentes selon les auteurs. Cette étape de correction de la pente peut être effectuée avant ou après la correction de l'inclinaison. Il est aussi possible d'effectuer une correction en trois temps : corriger la pente, corriger l'inclinaison et corriger la pente de nouveau.



(a) Image avec une pente

(b) Image corrigée avec un angle de $-4°$

FIGURE 3.5 – Exemple de correction de pente sur une image, la ligne de base est indiquée en orange

3.3.3 Correction de l'inclinaison

La correction de l'inclinaison redresse les lettres d'une imagette (voir figure 3.6). En effet chaque scripteur incline plus ou moins les caractères dans un mot, supprimer cette inclinaison permet donc de réduire les différences intra-caractères, en fonction du style de l'écriture. La méthode proposée par [184] utilise les histogrammes des traits continus verticaux. L'hypothèse de base est qu'une image de mot est redressée lorsque dans une image le nombre de colonnes contenant un trait d'écriture continu est maximal, c'est à dire quand il y a un maximum de pixels continus verticaux dans l'image. L'image est déformée entre les angles $[-\alpha; \alpha]$ et le nombre de traits continus dans une image déformée H_a pour un angle a est calculé :

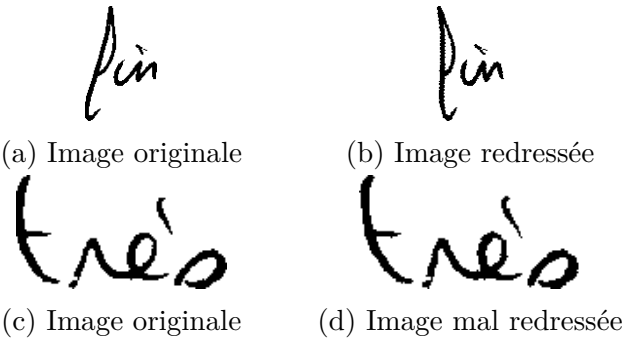
$$H_a = \sum_{m=1}^{M} \frac{h_a(m)}{\delta y_a(m)} \tag{3.2}$$

Avec $h_a(m)$ le nombre de pixels noirs d'une colonne m et $\delta y_a(m)$ la longueur du trait continu de cette colonne. Avec l'ensemble des H_a on obtient un histogramme représentant le nombre de traits noirs continus selon l'angle de déformation de l'image. L'angle maximisant le nombre de traits verticaux continus est retenu pour la correction de l'inclinaison.

Une méthode utilisant le code de chaîne est proposée par [122]. Similairement à la correction de la pente, [56] propose la transformée de Radon pour détecter l'inclinaison des caractères. Ce prétraitement peut aussi causer des déformations de l'image, comme dans le cas de la figure 3.6d. La maximisation du nombre de colonnes contenant un trait continu sur toute l'image n'a pas permis de redresser le caractère « t ». Au contraire en inclinant le « t » un plus grand nombre de traits continus est apparu. Bien que des détériorations peuvent apparaître, la correction de l'inclinaison améliore de manière générale la reconnaissance [141].

3.3.4 Normalisation de la position

La normalisation de la position recentre de manière similaire un ensemble de mots. L'objectif est de placer la ligne de base au centre de l'image de manière à réduire la variabilité intra-caractères. Si l'extraction de caractéristiques est indépendante de la taille, un recentrage de la ligne de base peut être suffisant. Dans d'autres cas il est nécessaire de redimensionner l'image [82]. Tout comme les étapes précédentes, la normalisation commence d'abord par une



(a) Image originale (b) Image redressée

(c) Image originale (d) Image mal redressée

FIGURE 3.6 – Correction de l'inclinaison par la méthode de Vinciarelli et Luettin [184]

détection de la ligne de base. Cette étape peut s'effectuer par une analyse de l'histogramme de densité horizontal [184].

Une fois les différentes étapes de prétraitements effectuées, les systèmes vont extraire des caractéristiques pour décrire un espace de représentation de l'image ou de l'imagette.

3.4 Caractéristiques

Les caractéristiques présentées au système de reconnaissance sont essentielles, de leur qualité dépendent les résultats finaux. Leur sélection est un problème complexe, il est possible de sur-dimensionner un vecteur de caractéristiques mais un espace de représentation de trop grande dimension peut avoir un effet néfaste sur la reconnaissance [25]. Les caractéristiques sont déclinées de plusieurs manières : selon l'échelle de la représentation, ou selon la génération de leur représentation. En classant les caractéristiques par échelle de représentation on distingue trois classes : bas niveau (information des pixels), moyen niveau (information structurée extraites des pixels), haut niveau (information structurelle sur l'image). Une seconde classification ne considère que deux types de caractéristiques : celles définies par le concepteur et les caractéristiques automatiques (définies par la machine). Nous utilisons cette seconde classification pour présenter différentes caractéristiques. Nous commençons d'abord par présenter les caractéristiques définies par le concepteur.

3.4.1 Caractéristiques définies par le concepteur

Partant d'une image en niveau de gris, le plus bas niveau de représentation est la valeur du pixel. Aucune caractéristique n'est extraite, le vecteur image

est fourni tel quel au système[84]. [81] propose des caractéristiques simples qui proviennent de l'information des pixels : moyenne des niveaux de gris, premier et dernier pixel noir, nombre de zones blanches entre ces pixels, etc. D'autres caractéristiques simples sont utilisées dans différentes approches : pourcentage de pixels noirs dans une zone [159], histogramme de densités horizontal ou vertical [46], moments invariants [91].

Différentes approches proposent l'utilisation des contours pour extraire des caractéristiques de moyen niveau. Les contours offrent une information de moyen niveau sur l'apparence d'un caractère. L'histogramme des directions du contour [102, 122] est une caractéristique relativement simple décrivant un contour. Chaque direction utilisée pour aller d'un point du contour à un autre est représentée par un encodage numérique, voir figure 3.7, dit code de Freeman [70].

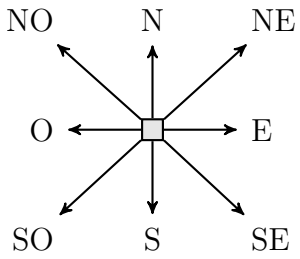


FIGURE 3.7 – Exemple d'encodage des contours

Dans sa première version elle est représentée par un histogramme des directions prises par le contour. [36] utilise des contours fermés pour en extraire une version améliorée du code de contour. La position, l'orientation et la courbure du contour sont calculées en plusieurs points pour rendre ces caractéristiques insensibles au bruit. L'approximation d'un caractère par des splines [163] est évaluée dans [178], encore une fois il s'agit d'une caractéristique basée sur le contour.

Les caractéristiques de haut niveau sont les plus humainement compréhensibles. Le nombre de boucles, présence d'une hampe ou d'un jambage dans l'image sont des exemples de caractéristiques de haut niveau. Ces caractéristiques sont généralement des méta-caractéristiques du niveau moyen. Une analyse du contour ou de la ligne de base est effectuée, puis des opérateurs sont utilisés pour déterminer la présence ou l'absence des caractéristiques de haut niveau. L'objectif principal de ces caractéristiques est de réduire la redondance dans une image en conservant les informations jugées discriminantes [105] par le concepteur.

On voit donc qu'il est très simple de construire un vecteur de caractéristiques à partir d'une imagette. La difficulté est de sélectionner les caractéristiques optimales par rapport à la tâche. [87] propose différentes méthodes pour la sélection de caractéristiques. Une Analyse en Composantes Principales [100][182][86] (ACP) peut par exemple être réalisée. Les caractéristiques corrélées apparaissent ainsi, leur recombinaison permet d'obtenir un vecteur final dont toutes les caractéristiques sont décorrélées. Un vecteur de caractéristiques défini par un humain est sans doute imparfait et il n'y a pas de certitudes quant à l'optimalité de cette représentation par rapport à la représentation idéale pour la machine.

3.4.2 Caractéristiques automatiques

Une façon différente d'obtenir des caractéristiques est de laisser le système les créer pour les besoins de la tâche. De nombreux algorithmes que nous présentons dans la section 3.5 produisent en interne une représentation des données qui leur sont fournies. C'est le cas par exemple des réseaux de neurones. Cette représentation interne des données peut tenir lieu de phases de génération de caractéristiques.

Cette approche est poussée à son extrême dans les architectures récentes des réseaux de neurones profonds [15], en particuliers pour les réseaux auto-encodeurs. Un réseau auto-encodeur est défini comme un réseau de neurones profond dont les poids sont réglés selon un apprentissage non supervisé dans un premier temps, puis supervisé dans un second, voir la sous-section 2.1.2. Durant la phase d'apprentissage non-supervisé, un réseau de neurone auto-encodeur génère une représentation interne des données qui lui sont présentées. Cette génération de caractéristiques a été explorée dans le domaine de la reconnaissance de la parole [47, 99, 162, 170].

D'autres approches non supervisées permettent de générer des caractéristiques. C'est le cas des algorithmes de partitionnement, notamment dans le cas de la génération de sacs de mots visuels [26, 152]. Bien que modifiant la représentation des caractéristiques en entrée, ce changement peut être intéressant dans certains cas. En effet utiliser une représentation simplifiée des données, ou modifier l'échelle du problème en réduisant ses dimensions, peut permettre de mieux discriminer les classes d'un problème. Cette modification de la représentation engendre généralement une diminution de la représentation en entrée, et donc une perte d'information. Cependant dans certaines applications cette perte n'est pas nécessairement importante. Les sacs de mots visuels sont utilisés

en classification de scènes [131], mais très peu en reconnaissance de l'écriture [152]. Cette différence de fréquence d'utilisation s'explique sans doute par la différence de taille des dimensions d'entrée avant l'étape sac de mots visuels. En effet dans le domaine de la classification de scène une image est décrite par des centaine ou des milliers de points, tandis qu'en reconnaissance de l'écriture les images de mots, ou les trames extraites par des méthodes de fenêtre glissante comprennent une dizaine à une centaine de points. La réduction de dimension, ainsi que la perte d'information liée, par sac de mots semble alors moins intéressante. Cependant il y a peu d'éléments de la littérature confirmant ces résultats.

Il est donc possible d'exploiter cette génération de caractéristiques pour la reconnaissance de mots [52, 65, 176]. L'un des avantages de générer automatiquement des caractéristiques est de ne pas injecter de connaissance a priori sur la tâche. Cela permet par exemple d'utiliser un extracteur de caractéristiques automatique pour la reconnaissance de l'écriture pour des tâches similaires [13, 27, 44](reconnaissance d'entités nommées, détection de mots). Le second avantage est de pouvoir enchaîner ces structures génératrice de caractéristiques pour créer des représentations d'un niveau d'abstraction croissant. Ainsi un premier générateur peut être utilisé pour produire des caractéristiques de bas-niveau, un second de moyen niveau, qui travaille à partir des premières et le troisième qui fournit des caractéristiques de haut niveau à partir des secondes. Ces caractéristiques une fois générées peuvent être passées au système de reconnaissance.

3.5 Reconnaissance

Cette partie du système traite le cœur du problème. Il s'agit d'utiliser des classifieurs performants optimisés pour la reconnaissance de l'écriture manuscrite. La reconnaissance de l'écriture peut se diviser en deux grandes tâches :

- la reconnaissance d'un caractère isolé indépendant des caractères suivants ou précédents, par exemple un chiffre, un symbole alpha-numérique ;
- la reconnaissance d'un mot ou d'une phrase, les caractères suivants et précédents ne sont pas indépendants les uns des autres et des règles doivent être utilisées lors de la reconnaissance.

Nous avons déjà présenté les classifieurs dans le chapitre précédent, nous présentons donc les deux classifieurs les plus utilisés.

Les Modèles de Markov Cachés (MMC), qui sont les modèles les plus connus

pour la reconnaissance de mots ou de phrases, disposent de nombreux cas d'utilisation dans la reconnaissance de l'écriture [14, 34, 61, 85, 110]. Ils sont utilisés pour plusieurs raisons, si nous nous référons au tableau 2.1 il s'agit d'abord de classifieurs dynamiques, ils sont donc adaptés à la reconnaissance de l'écriture qui est un problème temporel. Ensuite ils sont capables de segmenter une séquence et de l'étiqueter. Ils peuvent être utilisés tel quel [61], c'est à dire avec des mélanges de Gaussiennes, qui sont des modèles génératifs, ou bien les mélanges de Gaussiennes peuvent être remplacée par des classifieurs discriminants (réseaux de neurones, SVM) pour obtenir une meilleure discrimination au niveau local [14, 28, 149, 171, 176]. Ils sont généralement utilisés à deux niveaux :

— au niveau caractère, où chaque modèle représente un caractère, les états décrivant différents morceaux du caractère [61, 176] ;
— au niveau mot, où chaque modèle représente un mot, les états décrivant différents caractères ou morceaux de caractères.

Les réseaux récurrents « Bidirectional Long Short Term Memory and Connectionist Temporal Classification » (BLSTM-CTC) et les « Multi Dimensional Recurrent Neural Network and CTC » (MDRNN-CTC) sont les classifieurs état de l'art, ils possèdent les mêmes avantages que le MMC, mais en plus de cela ils sont discriminants, peuvent prendre en entrée un vecteur de grandes dimensions, et structure l'information spatiale de manière très efficace. Le système conçu est le système à l'état de l'art pour la reconnaissance de l'écriture [82, 85]. Il s'agit d'un réseau MDRNN-CTC, ce qui permet de garder un contexte temporel très important, tout en prenant des décisions très fortes lorsqu'un caractère est détecté. Cet ensemble couplé à l'intégration performante d'un modèle de langage permet d'obtenir des très bonnes performances sur des bases comme RIMES 2009.

Après avoir classifié une séquence de vecteurs de caractéristiques X en une séquence de sortie Y il est nécessaire d'effectuer une étape de post-traitements afin de corriger les erreurs liées à une mauvaise classification d'un vecteur de caractéristiques. Nous introduisons maintenant plusieurs de ces étapes.

3.6 Post-traitements

Les post-traitements permettent de corriger des erreurs effectuées lors de la reconnaissance. On parle de post-traitement, mais de nombreux algorithmes dits de post-traitements sont intégrés dans les algorithmes de reconnaissance,

ils agissent en tant que règles supplémentaires pour déterminer la solution. Cette étape est nécessaire car l'écriture manuscrite est un processus qui engendre beaucoup de bruit lors de la reconnaissance. Elle permet de corriger des erreurs minimes qui n'ont pas de sens au vu de la langue traitée, comme par exemple corriger « stode » en « stade » en français. La suppression des ambiguïtés peut se faire au niveau caractère pour modéliser des mots ou au niveau mot pour modéliser des phrases. Ce besoin de correction est lié à des difficultés à appréhender le contexte par les algorithmes de reconnaissance. Nous présentons trois méthodes de corrections fréquemment utilisées, en commençant par le décodage dirigé par le lexique, puis les dictionnaires et enfin les N-grammes.

3.6.1 Décodage dirigé par le lexique

Il est possible d'effectuer la correction en même temps que le décodage effectué par un classifieur. Par exemple à l'aide d'un décodage « time synchronous Viterbi beam search »[35, 54]. Ce décodage élimine un nombre de possibilités lors de l'exploration de l'algorithme de Viterbi. Le treillis n'est donc pas parcouru dans sa globalité, seules les hypothèses les plus probables sont conservées. Ce choix des hypothèses les plus probables peut se faire à l'aide d'un modèle de langage ou d'un modèle d'enchaînement des caractères ou d'un dictionnaire. Les hypothèses de décodage à l'instant t ayant une probabilité $p_t(x)$ inférieure à un facteur δ à $\max_x p_t(x)$ sont retirées du treillis. Le temps de calcul est ainsi réduit et le décodage est orienté vers les hypothèses les plus probables. La difficulté étant de régler le paramètre δ, trop grand il risque de ne pas filtrer les mauvaises hypothèses, trop petit il peut filtrer des hypothèses qui mènent à la solution recherchée.

Une seconde solution pour effectuer un décodage dirigé par le lexique est exposé dans [34], ce décodage est conçu pour effectuer un décodage au niveau phrase. Encore une fois il s'agit d'un algorithme dérivant de l'algorithme de Viterbi. À l'instant t, la probabilité d'émission d'un vecteur par un caractère et la probabilité que le caractère soit dans un mot w est défini comme

$$Q(t,(s,w)) = \max_{s'} \{p(o_t|s,w))p((s';w)|(s,w))Q(t,(s',w))\} \tag{3.3}$$

Avec $p(o_t|s,w))$ la probabilité d'émission de l'état $s \neq 0$ dans le mot w et $p((s';w)|(s,w))$ la probabilité de transition de l'état s' à s dans le mot w. Lorsque la transition se fait du dernier caractère d'un mot au premier caractère

d'un nouveau mot on a :

$$Q(t,(s=0,w)) = \max_{h} \{p(w|\mathbf{h}))Q(t,(S,\mathbf{h}))\} \tag{3.4}$$

Avec $s(0,w)$ le premier état du mot w et $(S,\mathbf{h})$ l'historique des mots construit à partir de modèles N-grammes de mots. Cette méthode cherche à définir le meilleur chemin après avoir pris en compte la séquence d'observation complète. Cette méthode bien que plus complexe en terme de mémoire (elle génère bien plus d'hypothèses), permet néanmoins d'utiliser un contexte très large pour corriger des erreurs locales.

Nous présentons maintenant des méthodes par dictionnaire qui sont plus rapides mais aussi moins performantes.

3.6.2 Dictionnaire

Le moyen le plus simple d'effectuer une correction consiste à utiliser un dictionnaire (ou lexique). Mesurer la distance entre un mot produit par la reconnaissance et les mots du dictionnaire, par une distance de Levenshtein [160] par exemple, permet de déterminer le mot du dictionnaire le plus proche et donc de corriger le mot. Ce dictionnaire est lié au contexte d'utilisation de l'algorithme de reconnaissance, il est donc nécessaire d'utiliser un dictionnaire adapté à la tâche. Dans le cas d'un vocabulaire très restreint l'utilisation des dictionnaires est très importante car elle permet de corriger une erreur de manière très simple. Par exemple dans le cas des chèques de banque, un dictionnaire comprend une trentaine de mots, avec des suffixes similaires (« soixante », « cinquante »). Dans [85] des performances avec des dictionnaires de différentes tailles sont présentés, on remarque clairement que sur les dictionnaires les plus restreints les résultats sont meilleurs. Dans le cas d'une phrase il est possible de contraindre la reconnaissance par la grammaire d'une langue [42]. De même le sens d'une phrase, ou son contexte dans un paragraphe permettent d'effectuer des corrections. Par exemple sur une ligne d'adresse postale le mot précédent la détection de « rue » est probablement un numéro tandis que le suivant est un lieu. La contrainte par un dictionnaire n'est cependant pas adaptée pour la reconnaissance d'entités nommés comme des noms ou des numéros de téléphone.

3.6.3 N-Grammes

Les N-grammes de caractères ou de mots définissent respectivement l'ensemble des enchaînements de caractères ou de mots possibles. Les transitions d'un élément à un autre sont probabilisées sur N pas. Par exemple la probabilité, dans l'hypothèse d'un modèle d'ordre 1, d'avoir la suite de mots $mots^W$ est défini comme :

$$p(mots^W) = p(mot_1) \times p(mot_2|mot_1) \dots p(mot_W) \times p(mot_W|mot_{W-1}) \quad (3.5)$$

Cela permet dans le cas de la reconnaissance de caractères d'obtenir plus de souplesse qu'un dictionnaire pour les mots hors-vocabulaire. La génération de ces modèles reste relativement complexe, en effet plus le pas N est grand et plus le nombre d'éléments est grand, et plus on risque d'obtenir des enchaînements rares. Par exemple si l'on considère l'alphabet latin uniquement avec les minuscules sur un pas de 2 on obtient 676 bi-grammes et sur un pas de 3 on obtient 17576 tri-grammes. Cette combinatoire pose des problèmes quand à l'optimisation des paramètres des N-grammes. Un modèle N-gramme peut être appris à l'aide d'un corpus de référence en utilisant la fréquence d'apparition de chaque N-gramme. Néanmoins un apprentissage aussi simple peut mener à des sur-représentations de certains éléments et à des sous représentations d'autres. Plusieurs méthodes comme le lissage, l'agrégation ou des modèles à maximum d'entropies [151] font partie des méthodes d'estimation qui permettent de déterminer les probabilités des N-grammes de façon plus pertinente.

3.7 Systèmes de reconnaissance

Dans les sections précédentes nous avons introduit des algorithmes individuels pouvant être intégrés dans une chaîne de traitements pour la reconnaissance de l'écriture manuscrite. Nous avons présenté cette chaîne de traitements comme pouvant être découpée en quatre parties : les prétraitements, l'extraction de caractéristiques, la reconnaissance et les post-traitements. Pour chacune de ces parties nous avons décrit différents algorithmes utilisés dans la littérature. Les chaînes de traitement, ou systèmes, sont généralement construites autour de l'algorithme de reconnaissance afin d'adapter le reste du système aux besoins (prétraitements et caractéristiques) ou aux résultats (post-traitements). Nous avons mentionné le fait que le cœur du système, l'algorithme de reconnaissance, force l'adaptation des autres parties afin de pouvoir fonctionner de

manière optimale. Par exemple les réseaux de neurones de par leur topologie figée dès leur création ne peuvent prendre en entrée que des vecteurs de caractéristiques de dimension fixe. Certains prétraitements ou certains vecteurs de caractéristiques permettent d'obtenir cette condition et il est donc nécessaire de les appliquer. Parfois ce sont aussi les besoins applicatifs qui forcent l'utilisation de certains algorithmes, par exemple dans le cas de la reconnaissance d'une séquence de caractères appartenant nécessairement à un vocabulaire il est important d'utiliser un algorithme intégrant un décodage dirigé par le lexique. Dans cette section nous présentons différents systèmes utilisant des algorithmes de reconnaissance différents, et travaillant sur différentes bases et nécessitant donc différents composants.

L'objectif est de présenter chaque système dans sa globalité, c'est à dire depuis les prétraitements jusqu'à la décision finale, afin de mettre en avant les forces et les faiblesses de chaque système. Nous détaillerons en particulier les caractéristiques et les algorithmes de reconnaissance qui sont intéressants pour notre travail. Après avoir mis en avant les forces et faiblesses des systèmes les plus récents nous soulevons des questions sur ces systèmes et proposons des pistes d'améliorations que nous allons explorer dans les chapitres suivants.

Nous présentons ces systèmes en les regroupant sous plusieurs grandes familles de systèmes afin de mettre en avant la force et les faiblesses de chaque famille. Nous décrivons trois grandes familles présentées dans la littérature destinée à la reconnaissance de l'écriture manuscrite. Ces trois grandes familles de systèmes sont nommées suivant les classifieurs sur lesquelles elles sont basées. Dans chaque sous-section plusieurs systèmes issus de ces familles sont décrits, et ce suivant un ordre chronologique. Nous présentons dans l'ordre les systèmes avec des Modèles de Markov Cachés, les systèmes neuro-markoviens et les systèmes avec des réseaux récurrents.

3.7.1 Systèmes avec des Modèles de Markov Cachés

Historiquement la première famille de systèmes permettant de réaliser une transcription complète d'une image de mots ou de ligne, les systèmes basés sur les Modèles de Markov Cachés[145] (MMC), restent encore très populaire puisqu'ils offrent une capacité de modélisation de la langue extrêmement efficace. De nombreux systèmes sont basés sur des MMC [18, 34, 35, 61, 109]. Les MMC offrent un cadre pour la modélisation de séquences temporelles complexes, ce qui se prête au problème de la reconnaissance de l'écriture manuscrite. La figure 3.8 présente l'architecture générale des systèmes MMC. Cette

figure introduit le concept de modèle d'attache aux données et de modèle des solutions recherchées. Le premier associe les données brutes avec les classes du problèmes, tandis que le second associe les différentes classes produites par le modèle d'attache pour former un mot ou un ensemble de mots. Le modèle des solutions recherchées peut s'appuyer sur des connaissances linguistiques ou des lexiques pour améliorer la décision au niveau mot ou phrase.

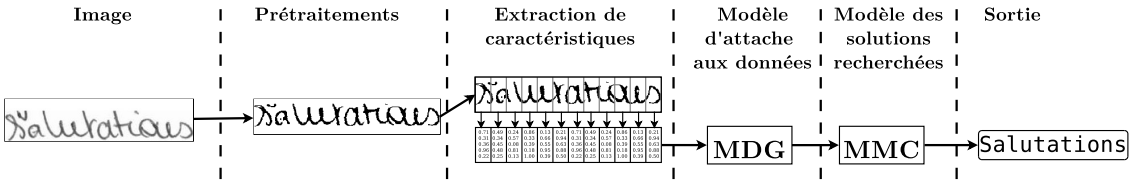


FIGURE 3.8 – Architecture de système MMC

Dans le cas du MMC les Mélanges De Gaussiennes (MDG) sont le modèle d'attache aux données et le MMC est le modèle des solutions recherchées. Nous présentons différents systèmes basés sur les MMC en suivant une évolution chronologique.

En 1994 Bunke, H.*et al.* [35] présentent un système dont la conception est très fortement liée aux images de mots présentées au système. La base concernée contient 12000 mots isolés manuscrits, avec 150 mots différents. Un mot possède 60 apparitions dans la base d'apprentissage et 20 apparitions dans la base de test. En plus d'avoir un vocabulaire relativement faible avec un grand nombre d'exemples du même mot, les images sont produites par 5 scripteurs qui ont eu pour règle d'imiter l'écriture d'un écolier suisse écrivant sur un repère horizontal afin de ne pas incliner l'écriture. La tâche considérée est donc relativement simple, ce qui leur permet de ne pas utiliser de correction de l'inclinaison des caractères qui peut être un prétraitement faisant perdre de l'information dans certains cas. La segmentation d'une image de mot est explicite et est effectuée par l'analyse des contours. Cette étape est relativement complexe et est décrite par un algorithme dont le but est de diviser le contour sur des nœuds du squelette de l'image tout en conservant un ordonnancement logique des différents segments. La figure 3.9 montre un exemple de cette partition sur un squelette d'image.

La représentation de l'image fournie au MMC est un mélange d'une représentation de haut niveau décrivant les contours (contour formant une boucle, nœud de contour, contour non bouclé) avec des caractéristiques de moyen niveau des contours (distances euclidienne entre différents points du contour). Les 52 classes (les 26 lettres de l'alphabet latin en minuscule et majuscule)

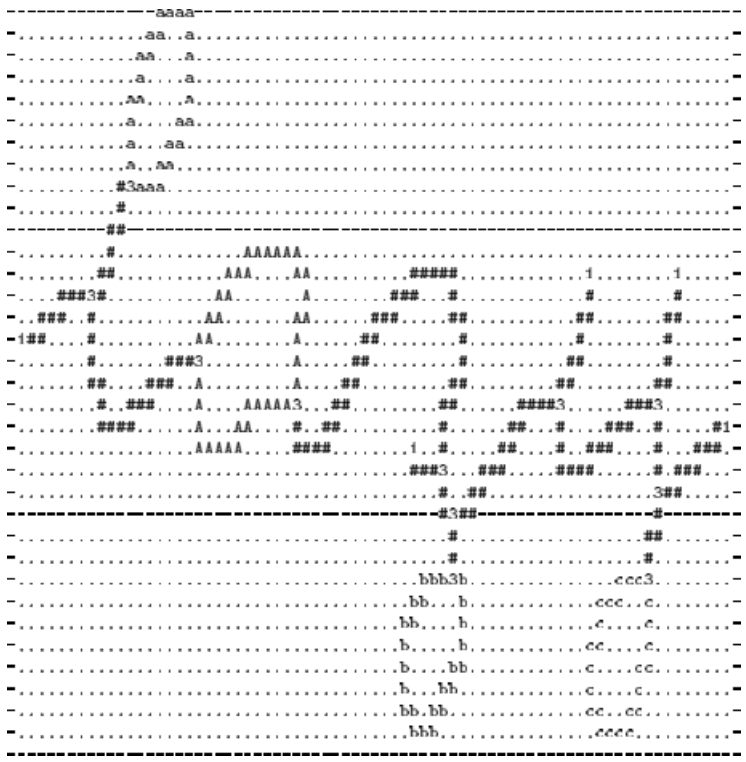


FIGURE 3.9 – Image d'un squelette de mot avec les différentes partitions des contours, les noeuds sont représentés par un « 3 », image provenant de [35].

sont chacune représentée par un MMC gauche-droite sans auto-transition et chaque mot est représenté par un MMC gauche-droite. Les MMC de caractères ont un nombre d'états variable, ce nombre d'état a été déterminé selon la segmentation produite pour chaque lettre et va de 1 état pour le « s » à 6 pour le « W ». Ce choix pour le nombre d'états est possible de par la faible variabilité de la base d'apprentissage. Les résultats obtenus sur la base sont évidemment excellent avec au maximum 3% d'erreur mot. Ces résultats ne sont pas intéressants d'un point de vue applicatif, les contraintes sur le système sont trop fortes pour qu'il puisse être utilisé dans des cas réels, mais ils permettent de mettre en avant la capacité des MMC à discriminer entre un symbole minuscule ou majuscule bien qu'ils soient parfois très proches graphiquement.

En 1998 Kundu H. *et al.* [110] proposent un système similaire avec une segmentation explicite et un modèle MMC non ergodique pour la modélisation des caractères. Dans leur application les transitions d'un état d'un caractère à l'état final d'un caractère sont autorisées.

Cette topologie de MMC permet de gérer une segmentation des caractères plus complexe. En effet dans le cas précédent l'écriture étant relativement stable la segmentation était régulière et très peu variable entre les différentes

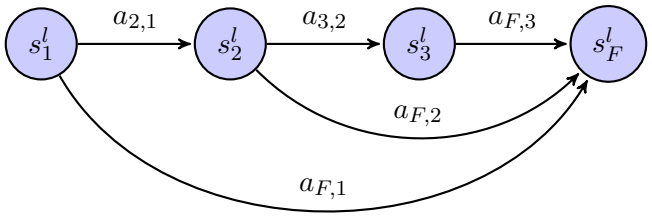


FIGURE 3.10 – Exemple de MMC pour le caractère « l »

apparitions d'un caractère.

Dans le cas d'une base produite avec moins de contraintes sur l'apparence des caractères la segmentation peut être très différente pour un même caractère. Les modèles non ergodiques permettent donc d'absorber cette variabilité de longueur de segmentation pour un même caractère. Dans le cas de la figure 3.10 le MMC représente le caractère « l ». Ce caractère possède peu d'états car il s'agit d'un caractère relativement court. Forcer la modélisation de ce MMC par un passage obligatoire par plus de 2 états risque de pénaliser la capacité du MMC à modéliser ce caractère, il est donc plus intéressant d'opter pour un modèle dont les probabilités de transition permettent de mieux modéliser la durée réelle du caractère.

Les expérimentations mises en place pour ce modèle sont plus proches du cas réel de reconnaissance de l'écriture, avec en apprentissage 21000 images de caractères isolés et plus de 3000 mots issus de courriers pour les tests. Les résultats montrent que ce modèle permet d'améliorer les résultats dans le cadre d'une segmentation explicite. Néanmoins cette méthode est plus gourmande en ressource qu'un modèle gauche-droite puisque le nombre de transitions est plus important.

El-Yacoubi *et al.* [61] proposent eux aussi un modèle de MMC plus complexe pour gérer des problèmes de segmentation de caractères. Ce modèle considère qu'un caractère peut être découpé au plus en trois segments par l'algorithme de segmentation explicite utilisé.

La topologie proposée pour les MMC caractères est composée de huit états, comme illustré sur la figure 3.11. Chaque transition possède une signification dans ce modèle. Par exemple la transition a_{06} représente le cas ou un caractère est segmenté comme un seul fragment (ce qu'ils appellent une bonne segmentation) tandis que les transitions a_{01}, a_{23} et a_{56} représentent le cas d'une sursegmentation du caractère et les transitions a_{12} et a_{45} représentent la nature de la segmentation (segmentation en deux ou trois parties). Il est important de

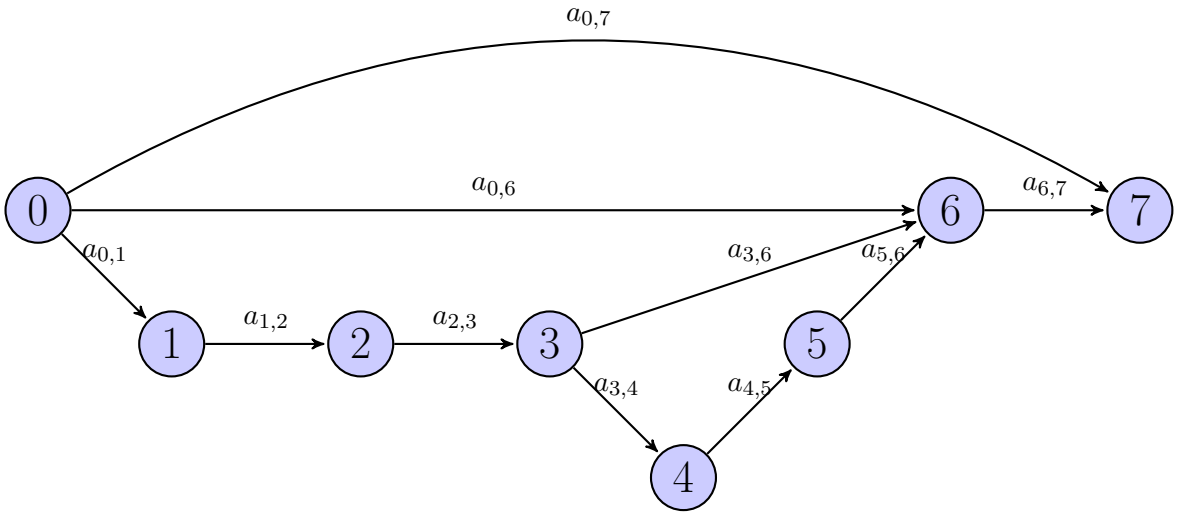


FIGURE 3.11 – Modèle MMC avec omission de transitions utilisé dans [61]

noter que les transitions a_{34}, a_{36}, a_{45} et a_{56} ont des poids partagés entre tous les modèles puisque selon les auteurs leur méthode de segmentation ne produit que très rarement une segmentation en trois fragments pour un caractère.

Les travaux présentés mettent aussi en avant une modification du modèle mot pour les MMC. Lors de la décision les modèles pour les caractères minuscules et majuscules sont mis en concurrence pour chaque caractère puisque le style d'écriture est inconnu. Les quatre probabilités de transitions majuscule-minuscule sont estimées sur la base d'apprentissage.

Les performances de ce système ont été mesurées sur une base de 11410 images de mots en apprentissage et 4313 images de mots en test provenant d'adresses de courriers. Le vocabulaire final est constitué de 9313 mots différents. Ce système réalise 99% de bonne reconnaissance avec décodage dirigé par le lexique réalisé avec 10 mots dans le lexique et 88.9% de bonne reconnaissance avec 1000 mots. Il est difficile de le comparer à d'autres approches étant donné qu'il n'a pas été testé sur des bases similaires.

Il est intéressant de constater que sur ces trois premiers modèles de MMC l'auto-transition sur un état n'est pas autorisée. Une sur-segmentation en quatre fragments d'un caractère ou une sous-segmentation de deux caractères fausse complètement le modèle au niveau mot (trop de caractères ou pas assez), ce qui requiert une segmentation explicite performante pour permettre au MMC de réaliser un alignement correct sur un mot.

En 2004 Bunke, H. *et al.* [34] mettent en avant un système de reconnaissance mettant en œuvre une segmentation implicite utilisant une fenêtre glissante

pour l'extraction d'une représentation de l'image. Les caractéristiques utilisées sont relativement simples, il s'agit de la densité de pixels noirs dans une sous fenêtre de chaque trame [183]. La trame est ainsi sous découpée en seize sous fenêtres, de dimension 4×4 pixels. Pour chaque sous fenêtre i la densité en pixels noirs est calculée :

$$f_i = \frac{n_i}{N}. \tag{3.6}$$

Avec n_i le nombre de pixels noirs dans la sous fenêtre i et N le nombre total de pixels noirs dans la trame. Ainsi le vecteur de caractéristiques représentant une trame est $f = (f_1, f_2, \dots f_{16})$. Cette représentation est très simple et peu coûteuse en temps de calcul. Elle permet aussi d'absorber une certaine variabilité de l'écriture : la densité de chaque sous fenêtre n'étant qu'une moyenne, elle lisse les irrégularités en évitant une description spatiale trop riche qui peut être sujette au bruit. De plus elle a l'avantage de comporter un très faible nombre de dimensions, ce qui lui permet d'être intégrée dans des méthodes de reconnaissance ne supportant pas les hautes dimensions, comme les MMC.

Cette représentation de l'information est ensuite fournie aux MMC. La fenêtre glissante va produire un nombre de vecteur de caractéristiques plus grand que le nombre de caractères, afin de gérer la segmentation d'un mot en une séquence de sortie de même longueur que la séquence d'entrée, la topologie choisie est un modèle gauche droite avec des auto-transitions, voir figure 3.12.

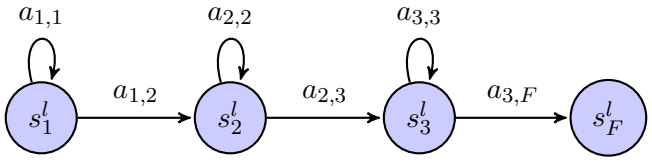


FIGURE 3.12 – MMC gauche droite avec auto-transitions pour le caractère « l »

Un caractère est modélisé par un MMC, cependant plusieurs hypothèses sont posées :

1. les différents caractères ont tous un modèle comportant N états, fixé à 12 lors des expérimentations ;
2. les minuscules et majuscules utilisent le même modèle, les caractères étant jugés similaires ;
3. seuls les caractères sont modélisés, les signes de ponctuation, les chiffres sont considérés comme du bruit.

Les modèles mots sont obtenus par concaténation des différents modèles de caractères. Les modèles sont appris à l'aide de l'algorithme de Baum-Welch, et le décodage est effectué avec l'algorithme de Viterbi.

Le décodage intégré pour la reconnaissance mêle étroitement la reconnaissance au modèle de langage. C'est un décodage orienté phrase ou ligne, basé sur l'algorithme de Viterbi. Il est modifié de manière à intégrer le fait qu'un caractère puisse être émis non seulement par les états, mais aussi par un mot existant. Lorsque le modèle mot est sur le premier ou le dernier caractère l'équation du décodage Viterbi est aussi modifiée de manière à intégrer le modèle du mot précédent ou suivant. Le meilleur chemin est identifié uniquement après avoir vu la séquence complète des observations. Le fait d'utiliser une segmentation implicite dans ce cas permet au contraire des travaux précédent de considérer que la segmentation en caractères est un produit de la reconnaissance et non pas l'inverse. Cela permet d'éviter des problèmes de mauvaise segmentation par un système explicite et donc d'être plus robuste aux variations d'une forme d'un caractère. En contrepartie le MMC comporte plus d'états et plus de transitions (en particulier avec les auto-transitions), l'apprentissage nécessite une plus grande quantité de données et la décision est plus longue.

Les performances de ce système sont mesurées sur la base IAM-DB et Reuters, les taux de reconnaissance mots sont respectivement de 45% et de 65% sur ces bases. Ces scores sont relativement stables pour des dictionnaires allant de 10000 à 50000 mots. La stabilité de cette méthode est principalement dûe à la modification de l'algorithme de Viterbi pour prendre en compte le contexte local et global.

Avec ce dernier système on voit un début d'évolution vers des systèmes moins dépendant d'une adaptation réalisée par un humain pour la base utilisée. L'évolution a lieu autant au niveau des caractéristiques, où l'on passe de caractéristiques de haut niveau compréhensible pour l'homme à des caractéristiques de bas niveau plus abstraites, qu'au niveau du système de reconnaissance où l'on passe de systèmes dont la reconnaissance est contrainte par la segmentation à des systèmes où la segmentation est un produit de la reconnaissance.

L'une des faiblesses non évoquée des systèmes basés sur les MMC, est leur aspect génératif. Comme expliqué dans le chapitre précédent les MMC utilisent des Mélanges de Gaussiennes pour générer la vraisemblance des observations étant donné un état. Les mélanges de gaussiennes peuvent cependant être remplacées par des réseaux de neurones qui sont des modèles statistiques discriminants. Nous allons maintenant présenter ces modèles et les évolutions qu'ils ont subi au cours du temps. Nous nous attachons particulièrement à présenter les évolutions des réseaux de neurones et leur apports dans des systèmes

neuro-markoviens.

3.7.2 Systèmes Neuro-Markoviens

Après avoir évoqué les systèmes basés sur les MMC seuls, nous décrivons maintenant les modèles hybrides neuro-markoviens. Nous avons d'abord décrit ces modèles dans le chapitre précédent 2.2.1.3, ils combinent l'aspect discriminant des réseaux de neurones au bas niveau tout en conservant l'aspect temporel au haut niveau. Bien que ces systèmes corrigent l'aspect génératif du MMC, ils ne corrigent pas les décisions prises indépendamment les unes des autres par le réseau de neurones.

Partant de ce constat H. Bourlard, et C. Wellekens proposent en 1990 [29] un système neuro-markovien (dans le domaine de la reconnaissance de la parole) avec un réseau de neurones intégrant la décision précédente dans le vecteur de caractéristiques présenté au réseau. Le vecteur d'entrée étendu V^+ à l'instant t est $V_t^+ = \{x_t, y_{t-1}\}$ avec x_t l'entrée du réseau à t et y_{t-1} la sortie du réseau à $t-1$. Cette topologie de réseau de neurones est similaire à celle présentée dans le chapitre précédent (section 2.2.3.1) à la différence que la récurrence est effectuée entre les entrées et les sorties du réseau. Cette architecture peut facilement être étendue pour intégrer un contexte temporel plus long sur les sorties. L'article propose aussi d'intégrer de la même façon le contexte sur les entrées. En généralisant le vecteur d'entrée devient donc :

$$V_t^+ = \{x_{t-n}, \ldots, x_t, y_{t-1}, \ldots, y_{t-k}\} \tag{3.7}$$

avec n le nombre de pas temporels conservés pour les entrées et k le nombre de pas temporels conservés pour les sorties. La figure 3.13 illustre le fonctionnement d'un tel réseau utilisant à la fois la sortie précédente et le contexte de la trame voisine.

Comparativement à un réseau de neurones dont la récurrence est effectuée dans une couche cachée, ce réseau garde intact le contexte temporel sur l'espace d'entrée et de sortie. Cette technique soulève cependant plusieurs problèmes sur l'initialisation du vecteur de caractéristiques lorsque les informations ne sont pas encore générées où ne peuvent pas l'être (sur des séquences trop courtes) et sur la véritable contextualisation de l'information. L'intégration du caractère précédent dans les caractéristiques peut mener à une mauvaise généralisation sur des vecteurs communs, en particulier les débuts de caractères, ou favoriser les bi-grammes de caractères les plus communs dans la base d'apprentissage.

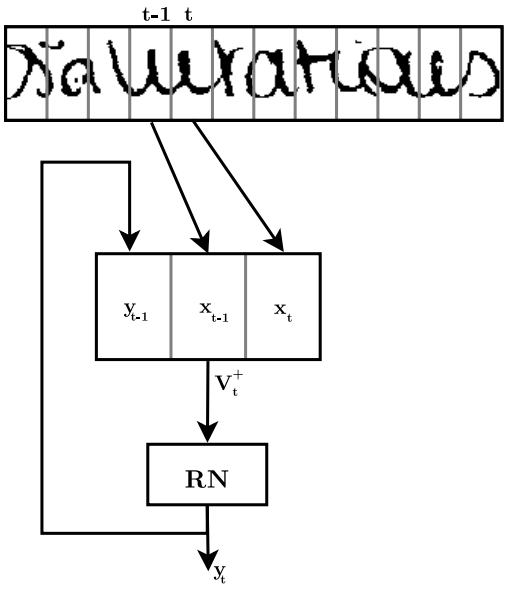


FIGURE 3.13 – Utilisation du contexte et de la sortie précédente dans un réseau de neurones

Les réseaux intégrant la récurrence dans la couche cachée semblent plus à même de généraliser l'information temporelle avec l'utilisation de leur mémoire interne.

L'approche de Bengio Y. *et al.* [14], présente un système neuro-markovien avec un apprentissage liant plus étroitement le MMC et le réseau de neurones. Le critère d'apprentissage est modifié afin d'intégrer l'aspect séquence et l'apport d'information venant du contexte du MMC. Au lieu d'appliquer un critère des moindres carrés, le critère d'Estimation de la Maximisation de l'Information Mutuelle (EMIM) est appliqué. Nous n'aborderons pas les détails mathématiques complets de cette méthode, mais l'utilisation de ce critère permet de calculer une fonction de coût par rapport à la matrice B du MMC qui permet par la suite d'estimer la correction des poids à effectuer sur le réseau de neurones. Il est important de noter que le réseau de neurones ne comporte aucune mémoire (il ne s'agit pas d'un réseau récurrent), alors que l'objectif de la méthode est de permettre au réseau de neurones d'avoir une cohérence temporelle des sorties plus importante. Il y a donc une forme de contradiction entre l'objectif désiré, obtenir un réseau de neurones fournissant des informations cohérentes au cours du temps, et le résultat produit : un réseau sans mémoire, dont seul l'apprentissage a été optimisé pour prendre en compte l'aspect temporel du signal traité. Le réseau est donc incapable de produire une suite cohérente de propositions de caractères. Cette optimisation a cependant son importance car elle permet d'améliorer les résultats dans le domaine de

la reconnaissance de la parole, puisque sur la même tâche le système passe de 87% de reconnaissance mot pour un système neuro-markovien non optimisé conjointement à 90% avec un système optimisé conjointement.

En 2009 P. Dreuw , P. Doetsch , C. Plahl *et al.* [57] comparent différents systèmes hybrides utilisant des réseaux de neurones mis en cascade afin de construire des vecteurs de caractéristiques de niveau de plus en plus haut après chaque réseau. Deux systèmes comparés dans cette publication sont constitués avec deux réseaux de neurones, le premier a pour but de réaliser une phase de prétraitement et d'extraction de caractéristiques à un niveau temporel très local (trame ou graphème de petite taille) tandis que le second assemble plusieurs sorties du premier afin d'apporter une décision à un niveau temporel plus large (plusieurs trames ou graphèmes). Le premier réseau reçoit les informations de l'intensité lumineuse des pixels à l'instant t et produit en sortie une classification sur l'alphabet concerné (dans cette publication il s'agit de l'alphabet arabe). Cette sortie est simplifiée par un log-ACP (log Analyse par Composante Principale) dans le but de réduire la complexité des sorties en rassemblant les éléments similaires. Le second réseau reçoit en entrée les sorties du premier réseau à l'instant t ainsi que les sorties à $t-p$ et $t+p$, où p est un pas temporel, avec en plus l'information de l'intensité lumineuse de $t-p$ à $t+p$. Cette forme est très similaire à un réseau de neurones convolutionnel, des caractéristiques de bas niveau sont concaténées sur un espace spatial de plus en plus large avant de prendre une décision au niveau caractère. Les sorties du second réseau sont identiques au premier et réduites par une log-ACP. Les MMC considérés pour les modèles de caractères comportent 3 états (quel que soit le caractère). Les résultats obtenus sur la base IfN/ENIT montrent que ce système hybride dépasse les performances des systèmes avec des MMC seuls. De plus la mise en cascade de réseaux de neurones permet d'améliorer les résultats de manière importante (8% d'erreur au niveau mot et 4% au niveau caractère) par rapport à un système sans cascade de réseaux, l'abstraction de l'information locale et son regroupement à un niveau plus important est donc intéressante pour la reconnaissance de l'écriture. Il semble que des unités temporelles de petite taille (graphèmes ou trames) ont besoin d'être rassemblées après avoir été abstraites pour décrire un ensemble temporel plus large mais plus cohérent. D'autres résultats intéressants sont mis en avant dans cette publication, par exemple l'importance de l'étiquetage initial des observations au moment de l'apprentissage : un mauvais étiquetage initial engendrera un mauvaise apprentissage des réseaux et une stagnation rapide des progrès du modèle

hybride. Ces travaux mettent en avant le besoin d'abstraire l'information à un niveau temporel et spatial avec des réseaux de neurones. Néanmoins dans ces travaux l'abstraction temporelle n'est pas réellement intégrée, il ne s'agit dans ce système que d'une concaténation de plusieurs abstractions spatiales.

Il est intéressant de voir que sur l'évolution des architectures neuronales on tend à vouloir s'abstraire des prétraitements, ou à les intégrer dans les réseaux [64]. L'extraction des caractéristiques est aussi de plus en plus déléguée aux réseaux neuronaux [176]. Cette abstraction des données initiales permet de mieux généraliser une architecture sur différents corpus. Si l'on compare le premier système MMC présenté [35] à un système neuro-markovien récent [18, 64] on remarque que le système a été adapté en fonction de la langue et non plus du corpus traité. Néanmoins à aucun moment un système neuro-markovien n'a réellement été capable d'introduire un lien entre la décision haut niveau et la décision bas niveau, les décisions restent toujours disjointes même si l'apprentissage a été réalisé conjointement.

Les deux systèmes présentés dans [18, 57] sont tous deux battus dans leur compétitions (ICDAR 2007 et ICDAR 2009) par des systèmes avec des réseaux récurrents. Dans le cas d'ICDAR 2009 il y a une différence de 2%, avec un dictionnaire à 100 mots, entre le système avec des réseaux récurrents et le second système avec un modèle hybride neuro-markovien. Cette différence passe à 6% avec un dictionnaire à 1600 mots et 8% avec un dictionnaire à 5000 mots. Ces systèmes permettent de corriger le problème majeur que nous avons montré dans les systèmes neuro-markoviens, à savoir l'indépendance des décisions des réseaux de neurones, tout en intégrant une abstraction temporelle et spatiale. Nous présentons maintenant ces systèmes.

3.7.3 Systèmes avec des réseaux récurrents

Les systèmes avec des réseaux récurrents se basent sur les développements les plus récents effectués dans le domaine [80, 82, 94]. Les réseaux de neurones récurrents existent depuis de nombreuses années [156, 192], mais jusqu'à de récentes améliorations sur leur algorithme d'apprentissage il ne leur était pas possible d'apprendre de réaliser un étiquetage des données lors de l'apprentissage. En 1996 A. Senior et T. Robinson [149] proposent un système avec un réseau récurrent utilisant un algorithme de décodage forward-backward pour estimer les probabilités de sortie de chaque caractère. Dans un premier temps afin d'initialiser le réseau récurrent les longueurs des caractères sont estimés de manière relative les uns aux autres. Ainsi dans une image un « l » occupe une

longueur relative $\frac{1}{2}$ tandis que le « m » ou le « w » occupent une longueur relative $\frac{3}{2}$. Cela permet d'estimer les probabilités de sorties de tous les caractères à tous les instants dans une image, c'est à dire d'obtenir un alignement forcé. Le réseau de neurones récurrent est appris dans un premier temps à l'aide de cet alignement forcé. Par la suite un décodage de Viterbi est effectué afin de proposer une segmentation capable de s'adapter aux variations de longueur des caractères dans l'écriture manuscrite. Après chaque itération d'apprentissage du réseau de neurones récurrent un nouvel alignement forcé est estimé avec un décodage Viterbi. Cependant le décodage Viterbi possède un désavantage majeur, il ne permet d'associer qu'un état à un instant donné et ne permet pas de créer une distribution sur l'ensemble des états. Cela ne permet pas par exemple de prendre en compte des zones d'incertitudes ou de transitions dans des mots (les ligatures). Une adaptation de l'algorithme forward backward est alors proposée afin d'obtenir une probabilité de distribution répartie sur l'ensemble des caractères. L'expérience montre que l'utilisation de l'algorithme forward backward est plus adapté pour la reconnaissance de l'écriture. Cependant cette forme de réseau n'est pas adaptée pour ce genre de problème à cause du problème de la disparition du gradient lors de l'apprentissage, ne lui permettant pas de prendre en compte un contexte temporel large.

En 2006 A. Graves [80] présente un apprentissage destiné au réseaux récurrents qui leur permet de décoder une séquence (l'apprentissage forward-backward du CTC, voir section 2.2.3.6) à l'aide d'un apprentissage similaire à celui présenté par Senior A. et Robinson T.. Cependant d'autres améliorations ont été apportées aux réseaux de neurones utilisés, en particulier pour corriger le problème de la disparition du gradient lors de l'apprentissage. Nous présentons deux systèmes état de l'art avec des réseaux de neurones récurrents, le second est une évolution du premier, généralisant les concepts à des espace à plusieurs dimensions.

Le premier système est issu des travaux dans [82], il s'agit d'un réseau de neurones dont l'apprentissage s'inspire très fortement des MMC et utilise des unités neuronales très performantes pour gérer la mémoire interne. Ce système intègre des prétraitements afin de normaliser l'apparence des caractères et de minimiser les variations entre les images. Les prétraitements effectués sont une correction de la pente, de l'inclinaison et une normalisation en hauteur.

À la suite de ces prétraitements une fenêtre glissante est passée sur chaque image pour extraire des vecteurs de caractéristiques de trames de 1 pixel de large. Afin de minimiser l'importance de la variation en hauteur, les caracté-

ristiques sont peu liées au positionnement absolu des pixels dans l'image et emploient des variations de ces caractéristique par rapport aux trames précédentes et suivantes. Ce vecteur de caractéristiques est composé de :

— la moyenne des niveaux de gris ;
— le centre de gravité ;
— le second moment vertical du centre de gravité ;
— la position des pixels noirs, le plus haut et le plus bas ;
— la variation de ces positions par rapport à la fenêtre précédente et suivante,
— le nombre de transitions pixel blanc/noir ;
— la proportion de pixels noirs entre le pixel noir le plus haut et le plus bas.

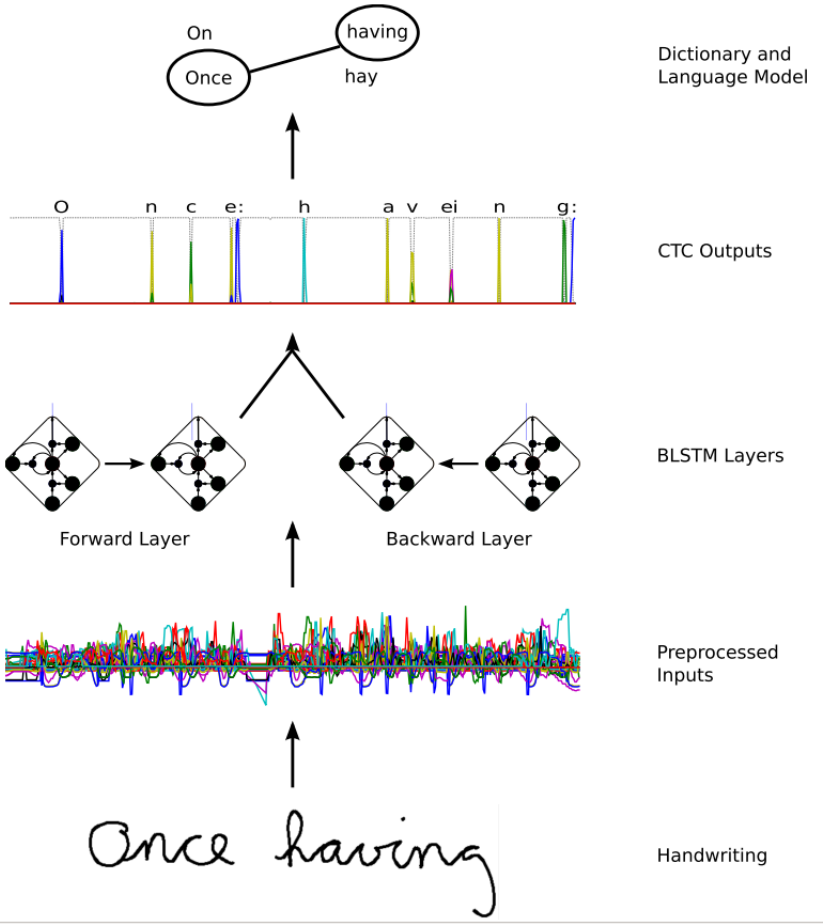


FIGURE 3.14 – Système BLSTM-CTC, image provenant de [82]

Ces caractéristiques sont ensuite exploitées par un « Bidirectional Long Short Term Memory – Connectionist Temporal Classification » (BLSTM-CTC) (voir section 2.2.3.6). Il s'agit d'une architecture avec deux réseaux de neurones

comportant des neurones LSTM. L'un des réseaux traite le signal dans le sens gauche-droite, l'autre dans le sens droite-gauche. Les informations de ces deux réseaux sont combinées par une couche CTC comportant $N+1$ label, le dernier label est un joker (absence de décision) ayant le même rôle que dans le système précédent : absorber le bruit et ne pas prendre de décisions à tous les instants notamment dans des situations ambiguës. Ce système de reconnaissance utilise donc un très large contexte passé et futur pour prendre une décision à un temps t. Il a une très forte capacité à absorber le bruit, dans la pratique il est observé qu'il ne se prononce sur des caractères que de manière très sporadique mais aussi de façon très fiable.

Un algorithme de token-passing est appliqué au niveau du CTC pour corriger les éventuelles erreurs du réseaux par rapport à un dictionnaire. Cet algorithme est adapté des MMC afin d'intégrer le lexique à la sortie produite par le système. Dans ce système le modèle d'attache aux données intègre des contraintes linguistiques : lors de l'apprentissage c'est le BLSTM qui apprend un modèle de langage puisqu'il intègre la mémoire du système et que les erreurs de classification d'une séquence sont propagées au travers de l'intégralité du réseau. La figure 3.15 illustre un système BLSTM-CTC.

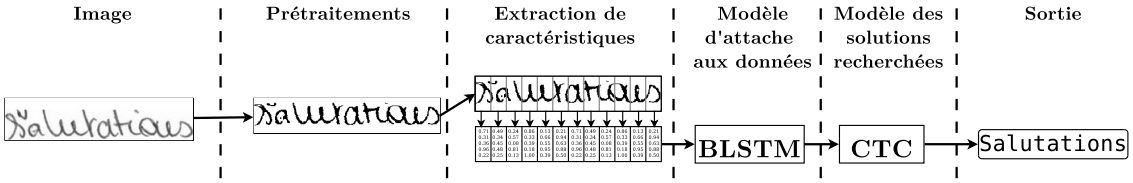


FIGURE 3.15 – Système avec des réseaux récurrents de type BLSTM-CTC

Le système suivant est issu des travaux présentés dans [84] . Il s'agit du système qui fait état de l'art dans le domaine de la reconnaissance des signaux multi-directionnels, « Multi-Dimensional Recurrent Neural Network » (MDRNN) . Ce réseau récurrent est une extension du modèle BLSTM à un signal 2D, le modèle BLSTM ne parcourant qu'un signal temporel 1D. Ce système n'intègre pas de prétraitements. Les caractéristiques utilisées sont l'intensité lumineuse des pixels de l'image et elles sont directement traitées par le MDRNN.

Les performances de ce système ont été mesurées sur la base IAM-DB avec 74.1% de reconnaissance mots avec un dictionnaire à 20000 mots. Sur la base RIMES les performances sont encore meilleures, avec 98% de reconnaissance mots avec un dictionnaire à 100 mots et 93.17% avec un dictionnaire à 1650 mots, respectivement les tâches WR1 et WR2 d'ICDAR 2009. Com-

Système	Reconnaissance en Top 1 (%)	Reconnaissance en Top 10 (%)
MDRNN-CTC	93.17	98.95
Système Neuro-Markovien	83.17	96.84
Système MMC	81.30	86.86

TABLE 3.1 – Résultats de différents systèmes sur la tâche WR2 de la compétition ICDAR 2009 [85]

parativement aux autres approches, utilisant des systèmes avec des MMC ou des systèmes neuro-markoviens ce système est bien plus performant comme le montre le tableau 3.1.

Sur la base arabe ICDAR 2007 ce système obtient les meilleures performances à savoir 91.43% de reconnaissance sur la base f et 78.83% sur la base s. Ce qui le place entre 3 et 4 points au dessus des autres systèmes, la plupart étant basés sur des MMC. Ces expériences montrent clairement la force de ce système, il n'y a plus de nécessité d'extraire des caractéristiques, le système est capable de les générer en interne afin de résoudre un problème complexe avec des données multidimensionnelles.

L'architecture générale est donc simplifiée par la disparition des prétraitements, comme illustré sur la figure 3.16

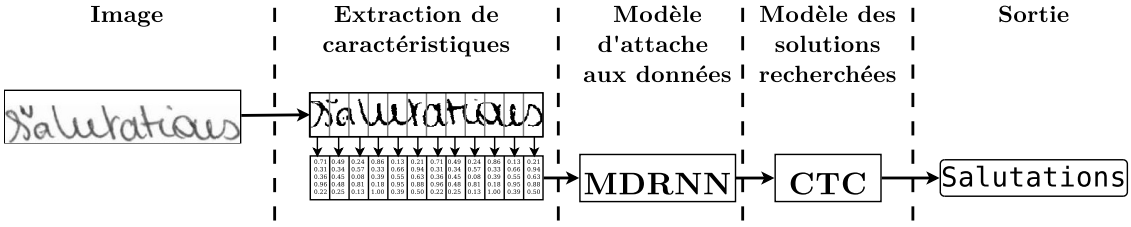


FIGURE 3.16 – Système avec des réseaux récurrents de type MDRNN-CTC

A2ia applique avec succès ce système lors de la compétition OpenHaRT 2013 [24]. Plusieurs prétraitements ont été appliqués (correction de l'inclinaison, normalisation en hauteur, etc.) mais aucun d'entre eux n'améliore les performances par rapport à un système sans prétraitements. Il n'y a pas de certitudes si cela vient de la stabilité de la base ou bien de la performance du système. Afin d'améliorer la convergence du MDRNN la méthode du « Curriculum Learning » [118] a été appliquée. Cette méthode d'apprentissage vise d'abord à entraîner le MDRNN sur des mots contenant peu de caractères puis à augmenter progressivement la taille des séquences de caractères à apprendre. Cet apprentissage progressif permet au MDRNN d'apprendre des caractères

isolés à l'aide des mots courts, puis d'apprendre les enchaînement de caractères à l'aide des mots longs. Cette méthode améliore légèrement les performances finales et elle accélère grandement la convergence du système. Le décodage est basé sur des modèles MMC pour les caractères avec un état par caractère puis des transducteurs à états finis [132] sont en charge du décodage mot. Les transducteurs prennent en compte le caractère joker du CTC. Ainsi un caractère peut transiter vers un autre caractère ou vers un joker. Le système ainsi produit atteint 18.4% de « Word Error Rate » (WER) sur cette compétition.

L'efficacité de ce système est incontestable, il combine une mémoire efficace avec une capacité de décision correcte extrêmement forte. Nous proposons d'analyser l'évolution des systèmes de reconnaissance de l'écriture ainsi que d'examiner des pistes d'améliorations pour ces architectures de réseaux récurrents.

3.7.4 Analyses et perspectives

Les sous-sections précédentes ont présenté trois familles de systèmes du domaine. Dans cette section nous proposons une analyse des évolutions de ces systèmes et à partir de cela nous présentons des perspectives pour notre travail.

L'évolution des systèmes de reconnaissance de l'écriture manuscrite montre que l'étude des prétraitements et de l'extraction de caractéristiques est de moins en moins utile. Tandis que les premiers systèmes devaient être adaptés manuellement au problème traité, aujourd'hui les algorithmes de reconnaissance sont capables d'apprendre par eux même la structure complexe contenue dans une image. Les systèmes avec des réseaux récurrents, et en particulier le système MDRNN [84], réalisent sans assistance humaine trois étapes des systèmes de reconnaissance qui étaient il y a quelques années encore considérées comme des étapes indispensables de la reconnaissance de l'écriture. Les performances affichées lors des différentes campagnes sur ce système montrent qu'il est extrêmement performant.

Il est intéressant de constater que certaines approches reviennent sur des systèmes neuro-markoviens tout en tirant profit des progrès réalisé sur les réseaux récurrents. C'est le cas du système développé par Hamdani M., Doetsch P., Ney H. en 2014 [89, 90] lors de la compétition OpenHaRT. Ce système combine un réseau récurrent composé de LSTM et un MMC. L'extraction de la représentation de l'image se fait par une méthode de fenêtre glissante, dans laquelle est extrait la valeur de l'intensité lumineuse des pixels après un repositionnement de la fenêtre basé sur le centre de gravité local. Le modèle de

MMC utilisé est un modèle gauche-droite avec des transitions additionnelles (appelé modèle de Bakis) entre l'état actuel et les deux états suivants, voir figure 3.17.

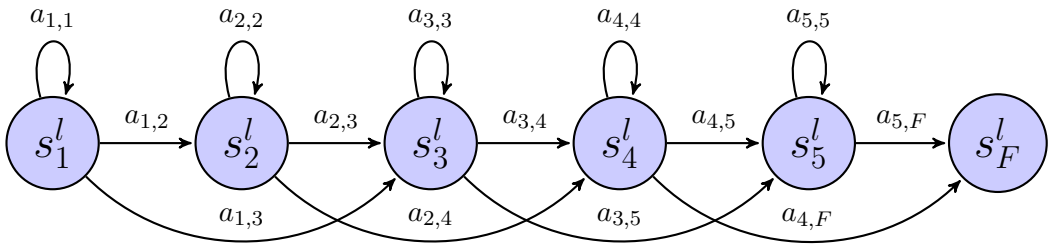


FIGURE 3.17 – MMC utilisé dans [90], modèle de Bakis

Ce modèle est plus souple qu'un modèle gauche droite simple puisqu'il permet de modéliser les caractères avec peu d'états de manière efficace sans pour autant pénaliser les modèles de caractères relativement long. Afin d'obtenir une mémoire efficace au bas niveau pour obtenir une décision caractère performante, le réseau récurrent LSTM vient remplacer les mélanges de gaussiennes dans le MMC. Deux avantages sont obtenus par cette combinaison LSTM/MMC. Premièrement les MMC sont plus adaptés que le CTC pour traiter des modèles n-grammes de grandes dimensions durant le décodage. Deuxièmement le fait d'apprendre un réseau récurrent LSTM non CTC sur la segmentation d'un MMC permet d'obtenir en sortie un signal moins piqué dans lequel plus d'alternatives sont proposées au niveau des caractères. Ce système neuro-markovien obtient des performances de 19.9% de WER et 8.3% sur la base de test d'OpenHaRT.

Plusieurs questions peuvent se poser au sujet des systèmes avec des réseaux récurrents. Premièrement, bien que la modélisation de la mémoire des neurones LSTM soit très performante, il est intéressant de se demander si des approches utilisant un contexte très large sur les observations, comme les CAC, ne permettent pas d'obtenir de meilleurs résultats. Nous allons pour ce faire explorer la piste d'un modèle hybride Champs Aléatoire Conditionnels (CAC) et MMC. Les CAC offrent un cadre d'étude statistiques entre le réseau de neurones récurrent et le MMC. De manière similaire aux réseaux de neurones récurrents ils utilisent le contexte temporel (à la fois sur les entrées et les sorties) pour prendre une décision au niveau caractère sur une trame ou un graphème. Le contexte séquentiel sur les sorties (ils sont limités à une dépendance d'ordre un) les rapproche des MMC, leur permettant ainsi de produire une séquence de sortie plus logique vis à vis de la langue étudiée.

Deuxièmement, comme nous l'avons expliqué les MMC sont plus adaptés que le CTC pour traiter des modèles n-grammes de grandes dimensions durant le décodage. Nous proposons donc d'utiliser un système combinant la performance au niveau caractère du BLSTM-CTC tout en appliquant un MMC pour le décodage dirigé par le lexique.

Troisièmement, la disparition des prétraitements et de l'extraction d'une représentation de l'information est elle raisonnable étant donnée la quantité d'algorithmes disponibles pour ces deux parties de la reconnaissance de l'écriture. Se passer des prétraitements semble intéressant car cela évite d'éventuels dégradation de l'image (dans le cas où le prétraitement échoue), cependant les caractéristiques développées au cours des dernières années dans le domaine de la vision par ordinateur apportent de plus en plus d'informations sur les événements bas niveau contenu dans une image.

L'inspection de ces trois points est l'objet du chapitre suivant, nous allons comparer un système CAC/MMC avec un système BLSTM-CTC en utilisant divers représentations de l'information.

3.8 Conclusion

Ce chapitre décrit un ensemble de techniques contribuant à la reconnaissance de l'écriture. Ce domaine a grandement évolué depuis ses débuts. Les premières recherches se sont concentrées sur les prétraitements et la représentation de l'information, puis avec l'amélioration de la puissance de calcul des ordinateurs le travail s'est transféré sur les algorithmes de reconnaissance et les post-traitements. Ce sont en particuliers le algorithmes de reconnaissance qui ont permis de grandes progressions dans le domaine. Ces algorithmes possèdent désormais une mémoire et une compréhension du contexte qui n'existait pas dans les premiers systèmes.

Dans ce chapitre nous n'avons pas présenté de systèmes complets, seulement des morceaux de systèmes. Ces systèmes complets sont généralement bâtis autour de l'algorithme de reconnaissance afin de prendre en compte les éventuelles faiblesses ou besoin de cet algorithme. L'algorithme de reconnaissance étant le centre du système, il est nécessaire parfois de s'adapter à ses besoins. Dans le cas des CAC nous avons montré qu'ils ne peuvent prendre en entrée que des données discrètes, il est donc nécessaire d'adapter le vecteur de caractéristiques à une représentation discrète. Dans le cas des réseaux de neurones, les connexions déclarées lors de la création du réseau fixent la

taille du vecteur d'entrée, il est nécessaire d'avoir un vecteur d'entrée de taille fixe. Cela signifie par exemple qu'une image doit être normalisée en hauteur pour obtenir un vecteur de caractéristiques d'une taille identique sur toutes les images traitées par le réseau de neurones. L'ensemble des algorithmes de reconnaissance nécessitent certaines adaptations par rapport à la finalité du système.

C'est ce que nous avons décrit dans la dernière section de ce chapitre qui décrit des systèmes de reconnaissance complets. Nous avons mis en avant le besoin qui a permis de faire émerger une famille de systèmes depuis la précédente : les systèmes MMC ont permis de classifier un signal monodimensionnel, les systèmes neuro-markoviens améliorent l'aspect discriminant, et les systèmes avec des réseaux récurrents lient le modèle d'attache aux données au modèle des solutions recherchées. Bien que la dernière famille de systèmes soit la plus performante à l'heure actuelle elle reste encore relativement méconnue et peu testée (en comparaison des deux autres). Nous proposons dans un premier temps d'explorer une piste alternative à ces systèmes et de la comparer sur des représentations similaires.

Chapitre 4

Comparaison de systèmes hybrides

Le chapitre précédent présentait différents systèmes de la littérature effectuant la transcription d'une image de mot, ou de ligne, en un texte numérique (une transcription). Ces systèmes, que nous avons classés en trois grandes familles de systèmes (Modèles de Markov Cachés (MMC), Neuro-Markovien et réseaux de neurones récurrents), possèdent des avantages et des inconvénients différents. Les systèmes MMC sont des systèmes avec un modèle génératif pour le modèle d'attache aux données, ce qui ne leur donne pas une bonne capacité de décision locale, c'est à dire sur une trame ou un graphème, néanmoins ils offrent des possibilités de modélisation de la séquence et de décodage assisté par le lexique. Les systèmes neuro-markoviens corrigent cet aspect génératif en remplaçant les mélanges de gaussiennes par un réseau de neurones. Néanmoins l'indépendance des décisions entre le MMC et le réseau de neurones ne permet pas d'utiliser l'information du contexte global pour décider localement, l'absence de retour d'information du MMC vers le réseau de neurones ne permet pas une décision locale pertinente. Les systèmes avec des réseaux de neurones récurrents corrigent ce problème en intégrant dans un réseau de neurones une mémoire contextuelle performante. Ces systèmes lient le modèle d'attache aux données au modèle des solutions recherchées, ce qui permet de corriger l'absence de retour entre ces deux parties de l'architecture. Cependant ces systèmes sont relativement récents et ne possèdent pas la capacité de décodage du MMC dans le cas de modèles n-grammes de grandes dimensions [90].

Dans un premier temps les contributions de ce chapitre se situent au niveau de la comparaison de différents systèmes appartenant aux familles présentées dans le chapitre précédent. Le premier système est un hybride neuro-markovien BLSTM-CTC/MMC. Le BLSTM-CTC génère une distribution de probabilité sur l'ensemble des caractères du problème, qui est ensuite utilisée par le MMC à la place des mélanges de gaussiennes. Le MMC a en charge la gestion lin-

guistique, nous intégrons le modèle de langage dans le MMC à la place du CTC, puisqu'il est plus adapté aux modèles n-grammes de grandes dimensions. Contrairement à un système neuro-markovien classique, l'alignement forcé permettant au réseau de neurones d'avoir une cible Y de longueur T à partir d'un mot Γ de longueur G durant l'apprentissage est réalisé par un algorithme inspiré du forward-backward mais adapté aux réseaux de neurones (voir section 2.2.3.6). L'alignement forcé évolue donc au fur et à mesure de l'apprentissage.

Le second système sur lequel nous contribuons est plus exploratoire, il s'agit d'un modèle hybride Champs Aléatoires Conditionnels (CAC)/MMC. Nous proposons de remplacer le BLSTM-CTC par un autre algorithme capable lui aussi d'utiliser un large contexte temporel, mais dont le fonctionnement est plus similaire à un MMC afin d'avoir des comportements cohérents entre le modèle d'attache aux données et le modèle des solutions recherchées. Ce système nécessite cependant un alignement forcé réalisé par un MMC au moment de l'apprentissage. Il est donc plus sensible à la qualité de l'alignement forcé initial.

Nous comparons ces deux systèmes à deux autres systèmes de références basés sur des travaux antérieurs. Nous comparons un système MMC et un système neuro-markovien réseaux de neurones/MMC.

Outre l'objectif de comparaison des architectures même, nous souhaitons évaluer les caractéristiques et les représentations fournies aux systèmes. Par ce terme nous entendons les caractéristiques fournies au système de reconnaissance, et surtout l'espace de représentation utilisé (booléen, discret, continu). En effet d'un côté le BLSTM-CTC a la capacité d'utiliser des entrées continues, de l'autre le CAC ne peut utiliser que des données discrètes. Nous comparons donc plusieurs caractéristiques de la littérature pour chaque système, ainsi qu'une adaptation des caractéristiques pour le système CAC/MMC. Nous proposons donc deux représentations des caractéristiques l'une continue, l'autre discrète.

Nous commençons par présenter les deux systèmes étudiées, puis nous présentons les résultats obtenus.

4.1 Systèmes de reconnaissance de l'écriture

Dans cette section nous détaillons les deux systèmes que nous comparons au travers de ce chapitre. D'un côté nous avons un système neuro-markovien

utilisant une représentation continue de l'information et de l'autre un système CAC/MMC utilisant une représentation discrète de l'information. Ces systèmes sont néanmoins identiques sur les prétraitements, les caractéristiques et le modèle des solutions recherchées. Nous décrivons d'abord une vue d'ensemble de ce système puis nous détaillons chaque partie.

4.1.1 Vue d'ensemble

Dans le but de comparer deux modèles d'attache aux données différents et des représentations différentes nous utilisons deux systèmes très similaires. L'architecture générale des deux systèmes est similaire à celles présentées dans le chapitre précédent : prétraitements, extraction de caractéristiques, modèle d'attache aux données et modèles des solutions recherchés.

L'image d'un mot est d'abord prétraitée afin de retirer le bruit et de normaliser l'apparence des caractères. Des caractéristiques et une représentation de l'information sont ensuite produites de l'image via une segmentation implicite par fenêtre glissante : une fenêtre de longueur P extrait une trame de l'image, cette trame est ensuite analysée et transformée par des opérateurs en un vecteur de caractéristiques de dimension M. Nous présentons quatre opérateurs d'extraction de caractéristiques et deux représentations différentes pour ces caractéristiques. Ainsi une image 2D de longueur T est transformée en une séquence $X = \{x_1, x_2, \ldots, x_t, \ldots, x_T\}$ de T vecteurs de caractéristiques $x_t = (x_t^1, x_t^2, \ldots, x_t^N)^\intercal$ de dimension N.

La séquence X est ensuite traitée par un modèle d'attache aux données qui transforme ce signal de longueur T et de dimension N en une séquence de sortie $Y = \{y_1, y_2, \ldots, y_t, \ldots, y_T\}$ de longueur T, avec $y_t = (y_t^1, y_t^2, \ldots, y_t^K)^\intercal$ de dimension K, où chaque y_t^k représente un caractère appartenant $c_k \in C = \{c_1, c_2, \ldots, c_k, \ldots, c_K\}$. Nous comparons deux philosophies différentes pour ces modèles d'attaches aux données. D'un coté le BLSTM-CTC qui utilise une mémoire du contexte, de l'autre le CAC qui utilise le contexte global de l'image et possède la connaissance de la décision précédente.

Quel que soit le modèle d'attache aux données, des erreurs apparaîtront dans les probabilités de sorties, ce qui mènera à une mauvaise reconnaissance d'un mot ou d'une phrase. Certains caractères pourront être remplacés par d'autres, certains peuvent disparaître et d'autres peuvent apparaître. Afin de corriger les erreurs de transcription produites par le modèle d'attache aux données la séquence est ensuite traitée par un MMC qui effectue un décodage dirigé par le lexique. Ainsi en s'appuyant sur un lexique des mots autorisés nous

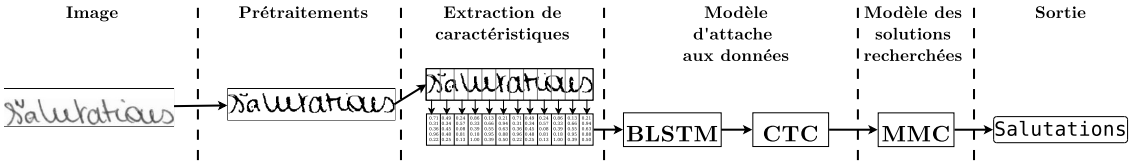


FIGURE 4.1 – Système BLSTM-CTC/MMC

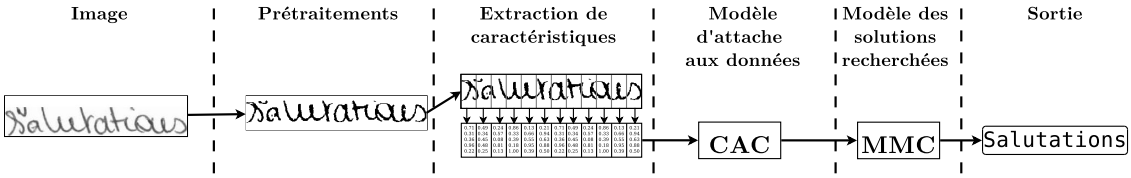


FIGURE 4.2 – Système CAC/MMC

pouvons corriger des erreurs de transcription. Dans certaines publications [81, 82] le BLSTM-CTC peut intégrer une étape de décodage avec un algorithme de Token Passing. Nous n'intégrons pas de décodage dans le CTC étant donné l'efficacité du modèle MMC pour effectuer cette tâche et afin de permettre une comparaison entre nos différents systèmes.

La figure 4.1 illustre le système BLSTM-CTC/MMC tandis que la figure 4.2 illustre le système CAC/MMC. Nous détaillons maintenant chaque partie de ce système.

4.1.2 Prétraitements

Afin de réduire la variabilité entre les images nous appliquons un ensemble de prétraitements communs aux deux systèmes. Nous appliquons les prétraitements classiques suivants : binarisation, correction de l'inclinaison, normalisation en hauteur.

Les informations en niveaux de gris permettent de connaitre la force du tracé, cette information ne nous est pas utile pour la reconnaissance de l'écriture, nous simplifions donc l'image en la binarisant. Nous appliquons une binarisation par seuillage fixe (seuil commun à toutes les images) puisqu'elle montre des performances plus intéressantes sur des mots isolés tandis que des binarisations plus complexes (Sauvola, Otsu) sont plus performantes sur des documents complets avec des ombres portées sur de larges zones de texte.

Nous corrigeons ensuite l'inclinaison pour deux raisons : la première est la réduction de la variabilité, et la seconde est de permettre la recherche pertinente de la ligne de base lors du prétraitement suivant. La correction de l'inclinaison est la méthode présentée dans [184] et en section 3.3.3.

Le BLSTM-CTC ne peut prendre en entrée que des vecteurs de dimension fixe, il est donc nécessaire d'assurer cette condition pour nos différents ensembles de caractéristiques. La normalisation en hauteur est effectuée afin que l'image entre dans un gabarit de hauteur définie, avec des dimensions prédéfinies pour la ligne de base, les hampes et les jambages. Ce prétraitement nous permet d'assurer cette dimension fixe pour l'ensemble des caractéristiques. Le choix des dimensions pour la normalisation est réalisé suite à des mesures statistiques sur la base. La ligne de corps contient la majorité de l'information, nous minimisons donc les déformations de cette zone. Nous utilisons la méthode décrite dans [184] pour déterminer la position de la ligne de corps. La moyenne et la médiane de hauteur de la ligne de corps avant toute modification a été ainsi fixée à 30 px. Nous utilisons cette valeur pour normaliser la hauteur de la ligne de corps. Les hampes et jambages sont normalisés à des hauteurs de 17 px afin de rentrer dans le gabarit nécessaire pour certaines caractéristiques. La déformation de ces zones est plus importante, mais elle permet néanmoins de conserver le contenu informationnel (présence de hampe, de jambage, accents, majuscules). L'image après normalisation a une hauteur de $30 + 17 \times 2 = 64$px.

La figure 4.3 illustre des résultats de notre chaîne de prétraitements.

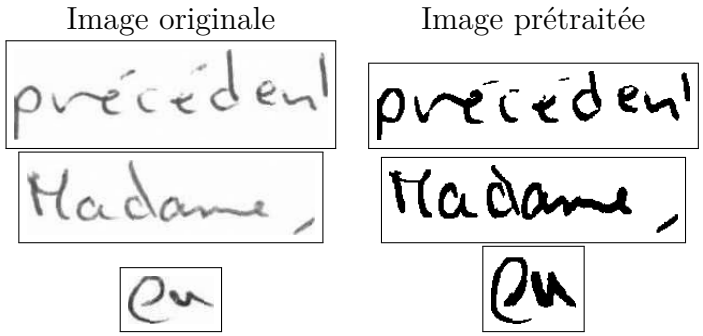


FIGURE 4.3 – Exemple d'images avant et après prétraitements

Après avoir transformé les images dans le but de réduire leur variabilité et de les adapter à nos besoins pour l'extraction de caractéristiques, nous utilisons des opérateurs mathématiques qui calculent une représentation numérique de l'information de l'image. Nous présentons maintenant ces opérateurs, ainsi que les représentations que nous obtenons à partir des ces opérateurs.

4.1.3 Caractéristiques et représentations

Après avoir réduit la variabilité entre les images, nous utilisons une méthode de segmentation implicite par fenêtre glissante pour extraire des informations

des images. La méthode de fenêtre glissante est décrite par deux paramètres, la longueur P de la trame (sous découpage de l'image) et la distance d entre chaque trame. Pour chaque trame nous utilisons des opérateurs, qui transforment une trame en un vecteur de caractéristiques. Nous présentons quatre caractéristiques utilisées dans nos travaux.

Nous comparons quatre caractéristiques différentes : les pixels, les Caractéristiques Des Pixels (CDP), les Histogrammes de Gradients Orientés (« Histograms of Oriented Gradients », HoG) et les « Oriented FAST and Rotated BRIEF » (ORB). Les trois premières caractéristiques ont été choisies pour leur efficacité démontrés précédente lors de précédentes utilisations dans le domaine de la reconnaissance de l'écriture [3, 82, 84, 176]. La dernière est présente dans un but expérimental. Nous comparons deux caractéristiques de bas-niveau (proche du contenu des pixels) et deux caractéristiques de moyen niveau (représentant des informations complexes extraits d'une transformation de l'image, comme l'image des contours ou l'image floutée). Afin d'éviter des représentations trop fortement liées à une langue nous n'utilisons pas de caractéristiques de haut-niveau.

On appelle représentation des caractéristiques la manière de décrire dans des espaces différents (booléens, discrets, continus, espaces clairsemés, etc.) une même caractéristique. Le modèle d'attache aux données du système neuro-markovien BLSTM-CTC/MMC utilise des représentations continues et celui du système CAC/MMC utilise des représentations discrètes.

Les caractéristiques présentées sont décrites dans un espace continu et ne sont donc pas utilisables directement par le système CAC/MMC. Nous utilisons dans ce cas un opérateur de discrétisation nous permettant de transformer une représentation continue en une représentation discrète adaptée pour le système CAC/MMC. Nous présentons les caractéristiques dans leur représentation continue, puis nous présentons la méthode de discrétisation de ces caractéristiques que nous utilisons.

4.1.3.1 Pixels

Nous utilisons cette caractéristique car il s'agit de la représentation naturelle de l'information contenue dans une image en niveaux de gris ou en noir et blanc. C'est une caractéristique de bas niveau qui a l'avantage de conserver l'ensemble de l'information sans risquer de la déformer. Cette caractéristique est importante car elle permet de montrer la capacité du modèle d'attache aux données à abstraire des connaissances à partir de la représentation du signal

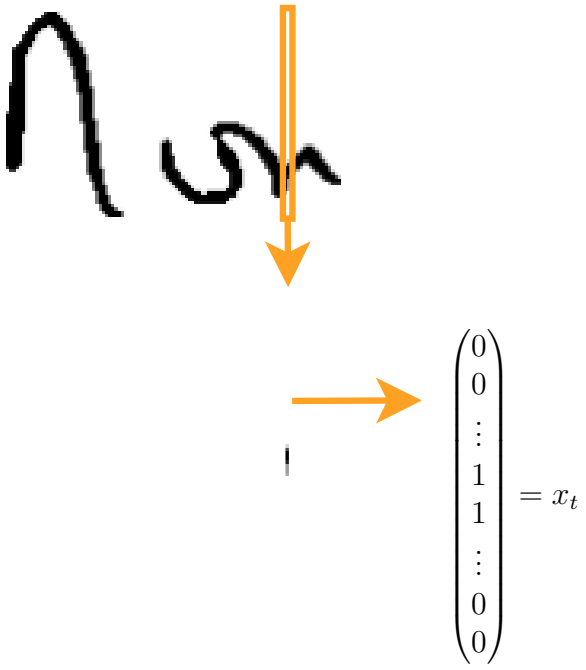


FIGURE 4.4 – Pixels

la plus élémentaire.

Pour chaque colonne d'une trame de longueur P nous extrayons l'états des pixels (noir ou blanc), le vecteur de caractéristiques constitué est de dimension $64 \times P$. Dans le cas des pixels nous avons choisi une trame de longueur $P = 1$ afin d'obtenir la représentation la plus élémentaire de l'image. Ce vecteur est illustré sur la figure 4.4. Le vecteur de caractéristiques x_t représentant les pixels est ensuite fourni au modèle d'attache aux données.

4.1.3.2 Caractéristiques des pixels

Les Caractéristiques Des Pixels (CDP) est le vecteur de caractéristiques proposé par A. Graves dans [82]. Il s'agit d'une compression de l'information contenue dans les colonnes de pixels, ce qui en fait donc une caractéristique de bas-niveau. Nous l'intégrons afin i) d'avoir une comparaison avec un système état de l'art, ii) de vérifier notre configuration du réseau BLSTM-CTC et iii) de mesurer la qualité de nos prétraitements.

Une trame de longueur $P = 1$ pixel est décrite par un ensemble de caractéristiques extrait via des opérateurs mathématiques :

— la moyenne des niveaux de gris ;
— la position du centre de gravité ;
— le moment vertical du second ordre des pixels noirs ;
— la position du pixel noir le plus haut et celle du pixel noir le plus bas ;

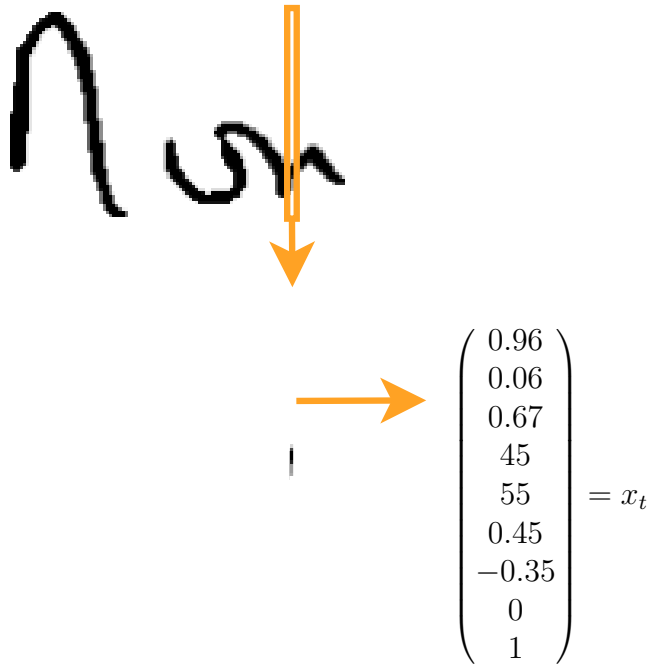


FIGURE 4.5 – Caractéristiques des Pixels

— la variation de cette position par rapport aux trames adjacentes ;
— le nombre de transitions noir/blanc entre le pixel noir le plus haut et le pixel noir le plus bas ;
— la proportion de pixels noir entre le pixel noir le plus haut et le pixel noir le plus bas.

Soit neuf valeurs décrivant une trame d'un pixel de long, comme illustré sur la figure 4.5. Le vecteur de caractéristiques constitué est ensuite traité par le modèle d'attache aux données.

4.1.3.3 Histogramme de Gradients Orientés

Les Histogrammes de Gradients Orientés [48] (« Histograms of Oriented Gradients », HoG) ont été employés dans le domaine de la reconnaissance de l'écriture dans [3, 116]. HoG est un descripteur de points similaire à « Scaled Invariant Features Transform » [119] (SIFT) ou « Speeded Up Robust Features » [10] (SURF). Un descripteur de points est généralement composé de deux parties : un extracteur de points [128], qui permet de repérer un point d'intérêt dans une image (point saillant) et un descripteur de points qui est un opérateur mathématique décrivant l'environnement autour de ce point saillant. Dans notre cas nous parlons du descripteur de point comme opérateur mathématique. HoG permet d'obtenir des caractéristiques de moyen niveau : les informations décrivent les contours de l'image et ne peuvent être interprétées

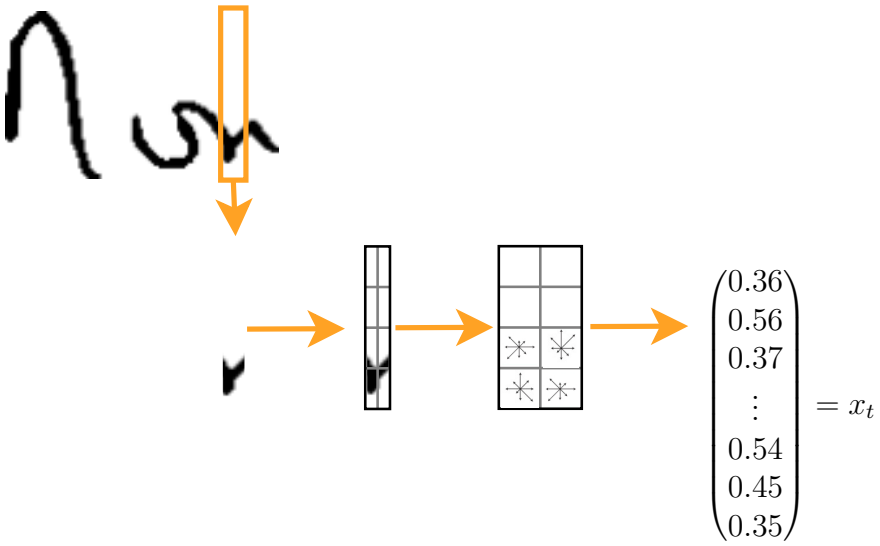


FIGURE 4.6 – Histogramme de gradients orientés

directement par l'homme. HoG a aussi été employé dans des domaines comme la reconnaissance faciale [51] où il se montre plus performant que SIFT, ou la détection d'objets dans une scène [48]

Les trames de l'image sont découpées en F sous-fenêtres I_f de dimension $n \times n$. Le gradient des directions de chaque sous fenêtre est calculé à l'aide d'un opérateur de convolution utilisant les noyaux suivants :

$$N_x = (-1, 0, 1) \text{ et } N_y = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix} \tag{4.1}$$

les dérivées selon l'axe x et l'axe y pour la sous fenêtre I_f est donc :

$$I_{f,x} = I * N_x \text{ et } I_{f,y} = I * N_y \tag{4.2}$$

L'orientation du gradient est donnée par :

$$\theta = \arctan \frac{I_y}{I_x} \tag{4.3}$$

Les orientations du gradient sont ensuite quantifiées en D directions, formant ainsi un histogramme de gradients orientés pour une sous fenêtre. Le vecteur de caractéristiques d'une trame est la concaténation des histogrammes des F sous-fenêtres de la trame. Le vecteur ainsi formé est donc de dimension $F \times D$.

Dans notre application nous utilisons des trames de dimension 8×64, découpés en 8 sous-fenêtres sans recouvrement de dimension 8×8. Le gradient est

quantifié en 8 directions, la figure 4.6 illustre la construction de notre vecteur. Ces valeurs ont été déterminées empiriquement. Le vecteur de caractéristiques est donc de dimension $8 \times 8 = 64$. Nous présentons maintenant le dernier vecteur de caractéristiques que nous utilisons qui est originaire du même domaine d'application, la vision par ordinateur.

4.1.3.4 ORB

« Binary Robust Independant Elementary Features » (BRIEF) [37] et sa variante « Oriented FAST and Rotated BRIEF » [153] (ORB), sont comme HoG, des descripteurs de points. ORB a l'avantage sur BRIEF d'être, comme SIFT, invariant à la rotation. ORB et HoG sont tous deux comparés à SIFT et SURF dans la littérature, mais jamais entre eux sur la même base. Ils sont tous deux plus performants dans des tâches de reconnaissance de scène [48, 97]. À notre connaissance la comparaison entre ces deux caractéristiques est inexistante dans le domaine de la reconnaissance de l'écriture manuscrite, ni d'ailleurs l'utilisation de ORB. Nous proposons donc d'utiliser ce descripteur de points dans un but exploratoire afin d'obtenir une comparaison ORB–HoG.

BRIEF est un vecteur de caractéristiques ou chaque caractéristique représente un ensemble de comparaisons d'intensités lumineuses (des tests) entre des paires de pixels provenant d'une image floutée. Un test τ entre deux points a et b est défini par :

$$\tau(p; a, b) = \begin{cases} 0 & \text{si } p(a) \geq p(b) \\ 1 & \text{si } p(a) < p(b) \end{cases} \tag{4.4}$$

avec p l'image floutée, $p(x)$ est l'intensité lumineuse d'un pixel. Le choix des pixels est aléatoire et suit une loi normale isotrope. Le vecteur binaire, construit via ces comparaisons à différentes échelles, a l'avantage d'être très rapide à construire et à utiliser. En effet la distance de Hamming est utilisée pour obtenir la distance entre deux vecteurs BRIEF.

Une caractéristique ORB est définie comme une combinaison de caractéristiques BRIEF :

$$f_n(p) := \sum_{i \leq 1 \leq n}^{I} 2^{i-1} \tau(p; a_i, b_i) \tag{4.5}$$

il s'agit du codage en entier d'une caractéristique BRIEF. Cette relation permet de passer d'une représentation de tests binaires à une représentation en entiers.

Tout comme HOG cette extraction de caractéristiques s'effectue dans plusieurs sous-fenêtres. Afin d'être invariant au redimensionnement, cette compa-

raison s'effectue à différentes échelles sur l'image.

Les résultats de [37] mettent en avant les performances de BRIEF sur ces prédécesseurs, SIFT et SURF, pour des tâches de classification de scènes. BRIEF, comme SIFT, SURF et HoG, est robuste aux variations de luminosité, robuste aux distorsions et invariant aux translations. En revanche il n'est pas invariant aux rotations.

ORB permet d'obtenir une variante de BRIEF invariante aux rotations en réalisant les comparaisons d'intensité lumineuses sur des images tournées à différents angles. Le vecteur de caractéristiques ORB de la trame est la concaténation des vecteurs ORB de chaque sous fenêtre, pour chaque angle de cette sous fenêtre et aux différentes rotations.

Dans notre application nous avons choisi de manière empirique 32 sous fenêtres de dimension 4×4 sans recouvrement, chaque sous fenêtre est tournée de 72 deg (soit 5 nouvelles images par sous fenêtre) et chaque sous fenêtre est redimensionnée 2 fois de suite avec un ratio de 1.5, soit 3 échelles au total. Cela constitue donc un vecteur de caractéristiques de dimension $32 \times 5 \times 3 = 480$.

Après avoir présenté plusieurs vecteurs de caractéristiques dont les valeurs sont réelles, nous présentons maintenant une manière de discrétiser ces valeurs afin de créer une nouvelle représentation de ces informations. Cette représentation est importante pour le système CAC/MMC car le CAC ne peut utiliser que des vecteurs de caractéristiques discrets.

4.1.4 Dictionnaire de mots visuels

Les caractéristiques présentées jusqu'ici peuvent être décrites dans un espace $\mathcal{R}^n$. Dans le cas du système BLSTM-CTC/MMC, le BLSTM-CTC prend en entrée une représentation continue des données (voir section 2.3), la représentation initiale de ces caractéristiques est donc adaptée pour ce système. En revanche dans le cas du système CAC/MMC, le CAC ne peut traiter que des données discrètes. Il est donc nécessaire d'utiliser un mécanisme permettant de transférer ces éléments d'une représentation continue vers une représentation discrète.

La représentation par Dictionnaire de Mots Visuels (DMV) permet de transformer un vecteur de caractéristiques en un élément d'un dictionnaire, c'est à dire qu'un vecteur décrit dans $\mathcal{R}^n$ est associé à un élément v_i du dictionnaire $D = \{v_1, v_2, \ldots, v_K\}$. La représentation devient alors discrète. Nous appliquons ce processus pour obtenir une représentation adaptée pour le système CAC/MMC.

À partir des caractéristiques extraites précédemment nous appliquons pour cela un algorithme de partitionnement standard, le K-means [43]. Les centroïdes ainsi produits par l'algorithme forment un dictionnaire de mots visuels. Ce dictionnaire est utilisé pour remplacer un vecteur de caractéristiques représentant une trame par son centroïde le plus proche. La figure 4.7 illustre un exemple d'extraction de dictionnaire qui peut ensuite être utilisé pour associer une trame à un centroïde. L'utilisation d'un dictionnaire de mots visuels transforme un espace de représentation $\mathcal{R}^n$ en un espace de représentation discret. L'efficacité de cette représentation est dépendante de la taille du dictionnaire : un dictionnaire trop petit engendre une perte d'information par rapport à la représentation initiale, un dictionnaire trop grand produit une distorsion de l'information initiale.

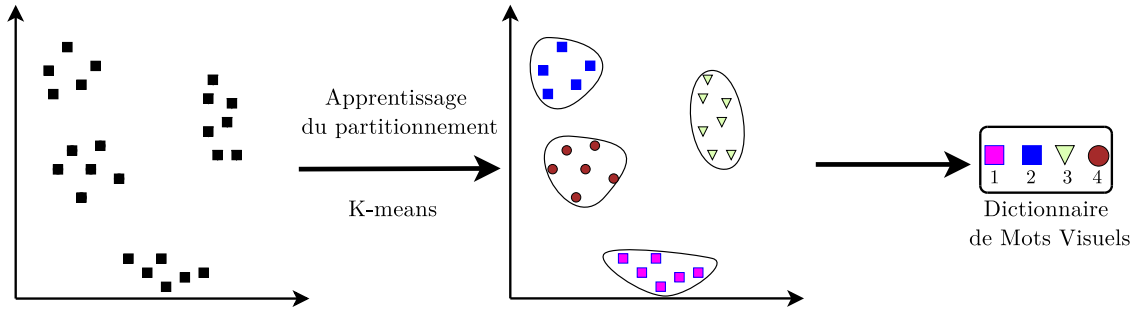


FIGURE 4.7 – Exemple d'affectation

Nous transformons donc un vecteur de caractéristiques d'une trame en un nombre appartenant à un espace discret. L'analyse au niveau d'une trame, représentée en figure 4.8, permet d'obtenir une information haut niveau pour chaque trame. Le partitionnement à ce niveau de granularité de l'image permet d'absorber certaines variations minimes d'une trame à une autre. Néanmoins, dépendant de la taille des trames, il peut être nécessaire d'avoir un dictionnaire de mots visuels de très grande taille. En effet plus les trames sont grandes plus le nombre de variations est grand, plus le dictionnaire doit être de grande taille pour prendre en compte ces variations.

Comme expliqué dans les sections 4.1.3.3 et 4.1.3.4, HoG et ORB sont des descripteurs de points dont nous extrayons l'information sur plusieurs sous fenêtres. Nous obtenons pour chaque sous fenêtre une description de cet espace, nous pouvons donc obtenir un partitionnement des données à un niveau plus fin pour ces deux caractéristiques. Cette granularité plus fine, représentée en figure 4.9 permet d'obtenir une information spatiale plus riche : la position de chaque fragment est conservée dans la représentation. Un autre avantage

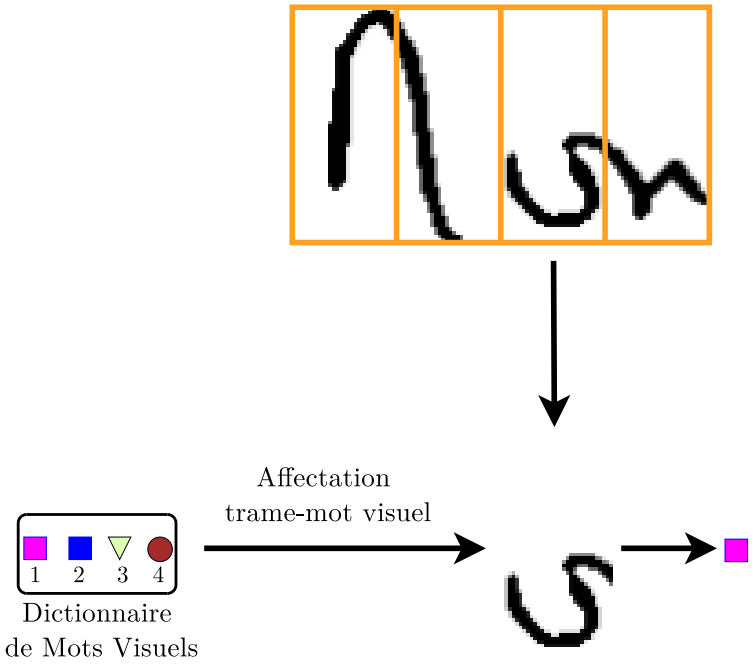


FIGURE 4.8 – Partitionnement des données au niveau trame

est qu'elle nécessite moins de partitions : les fragments sont décrits sur des espaces plus restreints présentant moins de variabilité. Cependant cette représentation est plus sensible aux variations minimes d'une trame à une autre, deux trames représentées par le même centroïde au niveau trames, peuvent avoir une représentation très différente au niveau des sous fenêtres.

Les quatre représentations présentées précédemment sont donc transformées à l'aide des dictionnaires de mots visuels adaptés, transformant des vecteurs de caractéristiques décrits dans $\mathcal{R}^n$ en vecteurs de caractéristiques discrets. Ce vecteur de caractéristiques discret peut être traité par le système CAC/MMC.

Après avoir présenté l'ensemble des représentations et caractéristiques explorées nous présentons les modèles d'attaches aux données qui utilisent ces informations pour prendre une décision au niveau caractère.

4.1.5 Modèles d'attache aux données

Nous comparons deux modèles d'attache aux données : le BLSTM-CTC et le CAC. Ces choix sont expliqués par les avantages, inconvénients et différences de ces deux classifieurs.

Le BLSTM-CTC possède une large mémoire temporelle grâce aux neurones LSTM et aux réseaux bidirectionnels, là où le CAC possède une utilisation du

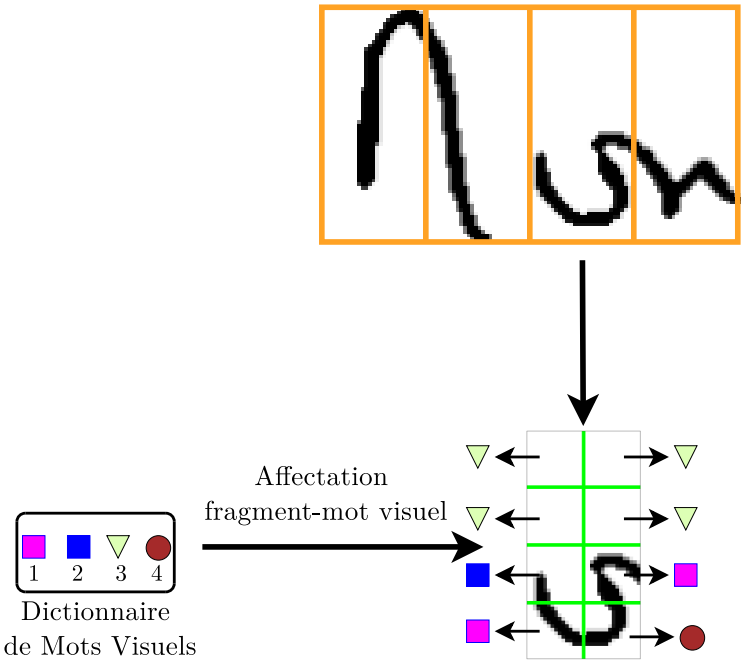


FIGURE 4.9 – Partitionnement des données au niveau des sous fenêtres

contexte sur les observations. De plus le CAC, de par la condition Markovienne, possède uniquement une mémoire de la sortie précédente. Il y a donc des différences importantes sur la dynamique des deux systèmes de classification.

L'apprentissage de ces deux classifieurs se fait par rapport à des sorties cibles différentes. Le BLSTM-CTC grâce à son caractère blanc et son apprentissage forward-backward est capable d'utiliser une séquence de caractères (un mot, une phrase) en tant que sortie cible. Il ne requiert donc pas un étiquetage des trames à l'aide d'un alignement forcé.

Le CAC ne possède pas un apprentissage capable de réaliser l'alignement entre la séquence d'entrée et la séquence de sortie. Il doit donc disposer de trames étiquetées pour réaliser l'apprentissage. Nous pouvons pour cela réaliser une annotation de la base via un alignement forcé généré par un premier système de type MMC [3]. Toutes les trames sont ainsi associées à une sortie cible qui est un caractère. Les résultats du CAC sont donc vulnérables aux erreurs commises par le MMC lors de cette phase préalable d'étiquetage.

Nous utilisons le logiciel RNNLIB [78] pour modéliser les BLSTM-CTC et CRFSuite [139] pour modéliser les CAC.

Les sorties du BLSTM-CTC et du CAC sont ensuite analysées par un MMC qui réalise un décodage dirigé par le lexique afin de passer outre les erreurs de reconnaissance de caractères pour trouver les mots du lexique les plus probables.

4.1.6 Modèle des solutions recherchées

Afin d'effectuer un décodage dirigé par le lexique, nous utilisons un MMC servant à la modélisation. Il est possible pour le BLSTM-CTC d'utiliser directement un algorithme de « Token Passing » afin d'effectuer un décodage dirigé par le lexique, mais étant donné notre objectif de comparer l'efficacité des deux modèles d'attache aux données d'une part, et l'efficacité supérieure des MMC pour décoder dans des lexiques de grandes dimensions d'autre part, nous avons choisi un système de décodage optimisé de type MMC identique pour les deux systèmes évalués.

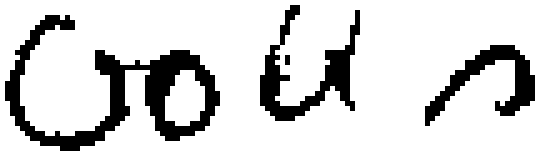
Les probabilités a posteriori des deux modèles d'attache aux données remplacent les vraisemblance des caractères générées par les mélanges de Gaussiennes, qui sont utilisées usuellement dans les MMC pour modéliser l'attache aux données. Le CAC produit en sortie de chaque trame des probabilités a posteriori pour chaque caractère qui peuvent être utilisées directement par le MMC.

L'emploi d'un joker par le BLSTM-CTC a un impact très fort sur ses sorties. Tout d'abord les sorties du CTC sont très sporadiques, mais très fortes : le caractère joker est la réponse par défaut du CTC tandis que les caractères sont des pics rares dans le signal de sortie. La figure 4.10b est un exemple de signal de sortie du CTC sur une image de mot dont on a extrait des caractéristiques HOG, les zones sans pics sont en réalité des plateaux du caractère joker. Le signal en sortie du CAC sur les mêmes caractéristiques est très différent, et ressemble plus à un ensemble de Gaussiennes.

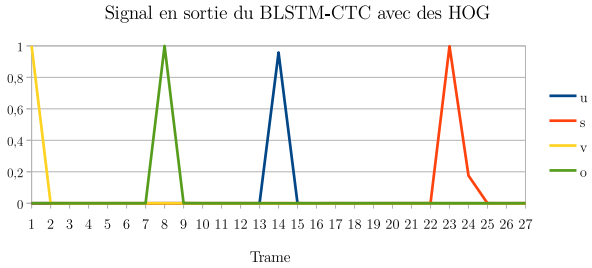
Le BLSTM-CTC est un classifieur dynamique qui se prononce très peu sur les caractères. Nous effectuons donc une étape de prétraitement de ces sorties, afin de retirer les zones de silence du CTC où les probabilités du joker sont très élevées. Nous retirons des séquences de sortie tous les vecteurs dont la probabilité a posteriori du caractère joker dépasse un certain seuil. Pour les autres trames nous éliminons aussi les probabilités du joker, les vecteurs sont ensuite normalisés afin d'être dans les bornes $[0; 1]$. Cette nouvelle séquence de sortie est alors fournie au MMC pour effectuer le décodage dirigé par le lexique.

Le MMC utilise un algorithme de décodage dirigé par le lexique de Viterbi intégrant un algorithme de Token Passing implanté dans le logiciel HTK [195]. La sortie produite par le MMC est alors un classement des mots les plus probables parmi ceux du dictionnaire fourni.

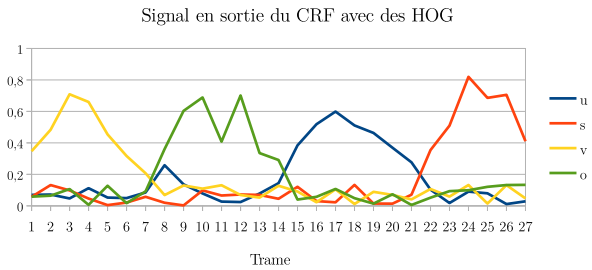
Après avoir présenté notre système qui transforme une image de mot en



(a) Image prétraitée du mot "vous"



(b) Signal de sortie du BLSTM-CTC



(c) Signal de sortie du CAC

FIGURE 4.10 – Image du mot "vous" et son signal de sortie du CAC et du BLSTM-CTC.

une sortie texte nous présentons les expériences menées.

4.2 Expériences

Après avoir présenté les deux systèmes, nous présentons maintenant l'expérience sur laquelle nous mesurons leur performance. Nous avons choisi une tâche de reconnaissance de mots isolés, qui nous permet de mesurer l'efficacité de ces systèmes dans un cadre pour lequel nous avons de nombreuses comparaisons à des systèmes existants [85]. Nous présentons dans un premier temps la tâche que nous effectuons, puis nous explicitons plusieurs paramètres que nous avons choisis pour notre système et finalement nous présentons les résultats obtenus.

4.2.1 Tache

Les modèles d'attache aux données et les représentations sont comparés sur la base RIMES mots [85]. La base RIMES est une base de documents factices rédigés à la main. Ces documents sont des lettres destinées à des administrations. Elles ont été produites par 1300 volontaires suivant des scénarios divers. Ces lettres sont rédigées sur du papier blanc puis scannées à une résolution de 300 dpi, ce qui en fait une base avec peu de bruit de fond. Quelques exemples de la base sont présents en figure 4.11, le ratio bruit sur signal est relativement faible. Les mots ont été isolés et annotés avec leur contenu exact.

FIGURE 4.11 – Exemples d'images de la base RIMES

La base est divisée en trois parties : apprentissage, validation, test. Chaque partie contient respectivement 44197, 7542 et 7464 images de mots. La figure 4.12 présente la répartition du nombre de mots par nombre de caractères sur les différentes bases. Cette répartition est très similaire entre les bases et est principalement centrée sur les mots courts, avec plus de 20% de mots de deux lettres dans la base.

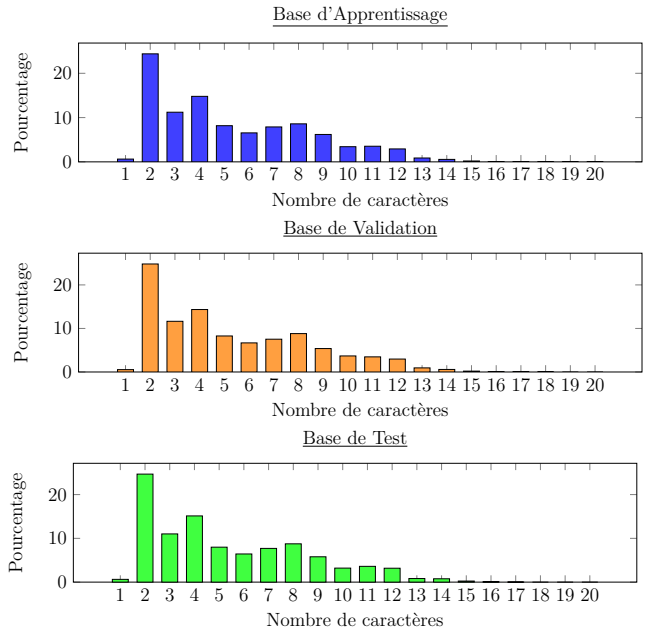


FIGURE 4.12 – Répartition du nombre de mot selon leur longueur

Les systèmes sont comparés sur la tâche $WR2$ de la compétition ICDAR

Caractéristique	Longueur de la trame (px)	Dimension du vecteur de caractéristiques	Nombre de mots dans le DMV
Pixels	1	64	128
CDP	3	27	128
HOG	8	64	512
ORB	8	480	1024
HOG sous-fenêtres	8	8	128
ORB sous-fenêtres	8	15	256

TABLE 4.1 – Paramètres communs entre les caractéristiques

2009 [85]. À partir d'une image de mot les systèmes doivent déterminer le mot recherché parmi 1600 mots choisis aléatoirement. Cette tâche est mesurée sur une liste de taille 1 (top 1) et sur une liste de taille 10 (top 10) produit par les systèmes. La réponse est considérée correct si la solution recherchée est dans la liste renvoyée. Comparer les représentations et les modèles d'attache aux données sur cette tâche permet la comparaison à plusieurs systèmes état de l'art [85].

4.2.2 Paramètres des systèmes

Nous avons testé différents paramétrages pour nos deux systèmes, que ce soit au niveau des pré-traitements, des caractéristiques ou des paramétrages du système de reconnaissance. Dans le tableau 4.1 nous résumons les hyper paramètres fixés pour les caractéristiques, puisqu'elles nous permettent de comparer nos deux modèles d'attache aux données. HOG sous-fenêtre et ORB sous-fenêtre désigne les caractéristiques pour lesquelles chaque sous fenêtre est considérée comme un élément indépendant dans le DMV. Concernant la longueur des trames nous l'avons optimisée sur la base de validation, ou suivant des utilisations reportées dans des publications. Nous avons donc des caractéristiques exploitant un contexte plus ou moins large et donc comportant un contenu informationnel sur le contexte plus ou moins important. Afin de ne pas perdre d'information entre les trames et de pouvoir apporter le plus d'information sur le contexte, les trames sont espacées de 1px quelle que soit la caractéristique. Le nombre de mots des dictionnaires de mots visuels a été optimisé sur la base de validation, de manière générale nous avons adapté la taille du dictionnaire à la diversité des composants que la caractéristique peut fournir, ce qui est lié à la fois à la taille des fenêtres mais aussi aux dimensions des vecteurs de caractéristiques.

Le CAC et le BLSTM-CTC ont 81 classes de caractères : 26 minuscules , 26 majuscules, caractères accentués en minuscule et certains en majuscule, dix chiffres, plusieurs symboles (tirets, pourcentage). Le BLSTM-CTC possède en plus de ces 81 caractères, la sortie correspondant au joker. La topologie du BLSTM-CTC a été fixée de manière à assurer une abstraction correcte au niveau temporel et spatial, il est donc nécessaire d'utiliser un nombre plus important de neurones LSTM sur la couche cachée que sur la couche d'entrée, de plus nous utilisons deux couches cachées BLSTM et des couches de sous-échantillonnage.

4.2.3 Résultats

Les résultats obtenus dans le cadre expérimental sont présentés dans le tableau 3.1. Les résultats sont présentés au travers de l'erreur mot (« Word Error Rate », WER) sur le Top 1 (meilleure hypothèse proposée par le système) et sur le Top 10 (dix meilleures hypothèses proposées par le système).

Nous comparons nos deux systèmes exploratoires à deux systèmes de références utilisant des caractéristiques HOG. Nous avons un système MMC avec quatre états par caractère et un système hybride réseau de neurones/MMC avec un état par caractère.

Système	Caractéristiques	Représentation	Top 1	Top 10
MMC	HOG	continue	36.25	20.35
MMC/réseau de neurones	HOG	continue	58.45	22.34
BLSTM-CTC/MMC	Pixels	continue	13.19	3.97
BLSTM-CTC/MMC	CDP	continue	13.06	4.34
BLSTM-CTC/MMC	HOG	continue	12.49	4.47
BLSTM-CTC/MMC	ORB	continue	**10.67**	**3.78**
CAC/MMC	Pixel	discrète (DMV)	75.12	69.4
CAC/MMC	CDP	discrète (DSMV)	65.34	59.4
CAC/MMC	HOG	discrète (DMV)	42.81	38.21
CAC/MMC	ORB	discrète (DMV)	54.09	50.37
CAC/MMC	HOG sous-fenêtres	discrète (DMV)	41.65	36.54
CAC/MMC	ORB sous-fenêtres	discrète (DMV)	52.05	49.35

TABLE 4.2 – Résultats des différentes représentations avec nos systèmes sur la tâche WR 2 de la compétition ICDAR 2009

Nous constatons de manière générale la supériorité du système BLSTM-CTC/MMC sur les autres systèmes et en particulier sur le système CAC/MMC.

Une différence si importante dans ces résultats peut être expliquée par les déficiences des autres systèmes, comme une déformation des caractéristiques ou une mémoire non adaptée pour traiter un problème sur de longues plages temporelles.

Tout d'abord comme nous l'avons mentionné il est possible que la discrétisation des caractéristiques pour le système CAC/MMC déforme le contenu de ces caractéristiques. Transformer un représentant d'un espace continu de dimension N en un représentant d'un espace mono-dimensionnel ne peut être effectué sans perte, cette transformation peut expliquer en partie cette différence de performance.

La qualité de l'étiquetage initial peut aussi être remise en cause. L'étiquetage initial permet au CAC d'avoir pour chaque entrée une sortie étiquetée au niveau caractère, si il se montre incohérent par rapport au contenu de l'image (par une segmentation imprécise des caractères), l'étiquetage peut détériorer l'apprentissage du CAC, de plus nous ne savons pas si un étiquetage effectué par un MMC convient au CAC.

Ensuite l'utilisation du contexte effectuée par le CAC, c'est à dire l'utilisation d'un ensemble d'informations de l'image pour prendre une décision à un instant t, peut ne pas être adaptée pour un modèle de modélisation de mots. En effet, bien que le contexte soit important, il est possible que le CAC ait appris à relier des caractères ou des morceaux de caractères qui ne sont pas nécessairement utiles. Il semble plus intéressant d'avoir une mémoire plus locale, comme dans le cas du BLSTM-CTC, qu'une utilisation de l'ensemble du contexte. Nous constatons ce phénomène sur les caractéristiques utilisant des trames de faible largeur (pixels et CDP), le BLSTM-CTC offre des résultats moins performants qu'avec des caractéristiques utilisant des trames plus larges mais néanmoins satisfaisant, tandis que le CAC subit une très forte baisse de performance.

Finalement après avoir observé les pondérations affectées aux transitions entre caractères dans le modèle CAC, nous constatons que la transition intra-caractères est souvent biaisée avec des poids trois à quatre fois plus importants que les transitions extra-caractères. D'une manière générale le CAC passe un temps plus important à transiter sur le caractère courant au risque de manquer des caractères courts (« i », « l », « t ») ou des caractères moins fréquents (caractères accentués). Bien qu'intéressant d'un point de vue de la modélisation linguistique, il semble que l'utilisation des transitions d'un caractère vers un autre desserve le CAC pour une tâche de reconnaissance de caractères dans

	Vérité terrain	Pixels	CDP	HoG	ORB
Vérité terrain	X	0.86	1.2	0.76	0.69
Pixels	0.86	X	0.95	1.2	1.1
CPD	1.2	0.95	X	1.18	1.09
HoG	0.76	1.2	1.18	X	0.8
ORB	0.69	1.1	1.09	0.8	X

TABLE 4.3 – Distances de Levenshtein entre les sorties textes générées à partir du top 1 des probabilités a posteriori des BLSTM-CTC

une image de mot.

Néanmoins il est important de préciser que d'autres travaux plus poussés sur un système CAC/MMC ont permis d'améliorer grandement les performances [20]. Le système proposé est similaire et utilise les caractéristiques HOG sur la même longueur de trames. Ces améliorations comprennent une prise en compte du contexte multi-échelle à l'aide de trois DMV ainsi qu'une amélioration du contexte local. Le premier DMV est construit de manière à discrétiser une trame de HoG, le second discrétise un ensemble de deux trames et le troisième un ensemble de trois trames, ces DMV ont respectivement 1000, 2000 et 5000 mots. Le contexte local est amélioré par la duplication de l'information des trames voisines, c'est à dire que chaque trame est décrite par le mot du DMV correspondant à cette trame ainsi qu'un nombre de mots décrivant les p trames suivantes et précédentes. Ce système atteint des performances de 31% de WER en Top 1 dans les mêmes conditions que nos expériences, ce qui n'est toujours pas suffisant pour égaler le BLSTM-CTC.

Concernant les résultats du BLSTM-CTC ils sont très satisfaisants et relativement intéressants. Quelles que soient les caractéristiques utilisées, les performances en Top 1 et Top 10 sont relativement similaires. Cependant il est intéressant de constater que les erreurs entre les BLSTM-CTC entraînés sur différentes caractéristiques ne sont pas identiques, comme nous l'illustrons au travers du tableau 4.4 et comme démontré par les distances de Levenshtein[160], produites entre les différentes sorties, dans le tableau 4.3.

Ce résultat est intéressant car il montre clairement que les caractéristiques apportent une abstraction de l'information différente et sont donc complémentaires. Ce résultat montre également que le BLSTM-CTC n'est pas capable de produire une abstraction similaire ou plus performante directement à partir des pixels. Proposer des caractéristiques complexes structurant l'information (ORB et HOG) à ce système reste donc relativement intéressant.

Si nous comparons nos systèmes BLSTM-CTC/MMC à ceux présentés lors

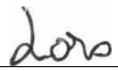
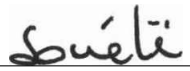
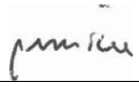
Image	Pixels	CDP	HoG	ORB	Vérité terrain
	dans	dans	**Lors**	donc	**Lors**
	doute	doute	compté	suite	**société**
	**d'assurance**	d'une	**d'assurance**	**d'assurance**	**d'assurance**
	par	quel	par	quel	**prend**
	**première**	croire	semaine	permis	**première**

TABLE 4.4 – Comparaisons de différentes sorties erronées du Top 1 des différents systèmes BLSTM-CTC appris sur des caractéristiques différentes

de la compétition ICDAR 2009 à l'aide du tableau 4.5, nous constatons que nos systèmes BLSTM-CTC/MMC sont positionnés juste derrière le système MDRNN-CTC avec token-passing [82].

Système	Top 1	Top 10
MDRNN-CTC (TUM)	6.83	1.05
BLSTM-CTC - ORB	10.67	3.78
BLSTM-CTC - HOG	12.49	4.47
BLSTM-CTC - CDP	13.06	4.34
BLSTM-CTC - Pixels	13.19	3.97
Système neuro-markovien (UPV)	13.89	2.05

TABLE 4.5 – Nos résultats comparés à ceux de la compétition ICDAR 2009 [85]

Le système MDRNN-CTC est le système état de l'art pour l'étude et la modélisation de signaux multi-dimensionnels. Appliqué à la reconnaissance de l'écriture il ne nécessite pas de prétraitements ni d'extraction de caractéristiques, il réalise de lui même ces opérations. Nos systèmes paraissent avoir des résultats très inférieurs, nous expliquons cela par différents points. Premièrement nous utilisons un système BLSTM-CTC, qui est un système mono-dimensionnel, ce système intègre moins de connaissances et effectue une abstraction de l'information moins importante qu'un système MDRNN-CTC qui est hiérarchisé en couches successives traitant différent niveaux d'abstraction. Deuxièmement étant donné qu'il s'agit d'un système mono-dimensionel nous

avons dû intégrer des prétraitements, hors la qualité de nos prétraitements n'est pas optimale. En effet nous n'avons pas optimisé les paramètres de nos prétraitements (seuil de binarisation, pas d'angle pour la correction de l'inclinaison, hauteur de l'image normalisée, taille de la ligne de base, ainsi que d'autres seuils interne à la normalisation) et nous nous sommes concentrés sur l'optimisation de la capacité de représentation de l'information de nos caractéristiques dans des conditions non-optimales.

Cependant nos réseaux BLSTM-CTC possèdent plusieurs avantages par rapport au MDRNN-CTC : i) leur apprentissage est rapide (un à deux jours sur la base RIMES au lieu de plusieurs semaines), ii) le nombre d'hyper-paramètres est moins important et moins complexe à maîtriser, iii) la convergence est difficile à maîtriser [118]. Nos réseaux BLSTM-CTC bien que moins performants offrent des performances voisines au MDRNN-CTC et supérieurs au systèmes hybrides. Ils sont donc intéressants d'un point de vue industriel, dans le cas où nous avons besoin d'une réponse rapide à un problème.

Nous concluons sur cette expérience en présentant des perspectives sur ces travaux.

4.2.4 Conclusion

Cette expérience menée sur une base de mots isolés montre clairement l'efficacité du système BLSTM-CTC/MMC par rapport au système CAC/MMC. Comparativement au CAC, le BLSMT-CTC a une mémoire plus efficace qui est capable de générer une abstraction du contexte temporel et spatial plus adaptée à la reconnaissance de l'écriture manuscrite.

Le système CAC/MMC est probablement pénalisé par le poids des autotransitions qui est trop important par rapport aux transitions d'un caractère vers un autre. Il serait intéressant de diminuer l'influence de ces transitions, soit pendant, soit après l'apprentissage des pondérations du CAC, afin de mesurer l'influence de ces transitions. Une autre modification possible afin de permettre au système d'utiliser des caractéristiques continues et donc d'éviter de déformer l'information contenue dans les caractéristiques en utilisant un DMV, est de remplacer le CAC par un CAC à états cachés [143, 169]. Celui-ci a la capacité d'apprendre une représentation interne de l'information, de manière similaire à la couche cachée d'un réseau de neurones. Ce système est néanmoins plus complexe à apprendre et requiert un temps d'apprentissage plus important. Une autre amélioration du CAC consiste en un apprentissage capable de générer le décodage durant l'apprentissage, de manière similaire à

un BLSTM-CTC ou un MMC. Cette amélioration peut s'effectuer en intégrant un algorithme de type forward-backward ou Viterbi lors de l'apprentissage. Les résultats de ce système ne nous ont pas motivé à améliorer ce système, nous avons préféré nous intéresser au BLSTM-CTC, pour tenter d'en améliorer les performances.

Le BLSTM-CTC nous a permis de produire des résultats intéressants à partir de caractéristiques, néanmoins ce classifieur n'est pas parfait et nous proposons des améliorations sur ce système. Nous avons constaté que les différentes caractéristiques produisent des résultats différents et donc complémentaires, nous pouvons donc imaginer des systèmes combinant les informations produites par les BLSTM-CTC pour améliorer les résultats globaux. Un autre problème constaté avec la combinaison BLSTM-CTC/MMC est lié au signal produit par le BLSTM-CTC et fourni au MMC. Le BLSTM-CTC produit un signal très piqué (voir figure 4.10b). Il se prononce peu sur les caractères, sur un temps très faible, avec une grande amplitude (probabilité proche de 1) sur le caractère détecté, et une amplitude quasi nulle sur les autres caractères (entre 10^{-9} et 10^{-15}). Les décisions effectuées au niveau caractère sont très (trop) tranchées avec des probabilités quasi-binaires, il n'y a que rarement des hésitations importantes entre plusieurs caractères. Ce qui signifie que dans le cas où le BLSTM-CTC effectue une erreur sur un caractère, le MMC risque de ne pas pouvoir déterminer le mot correct puisqu'il n'y a pas de caractères alternatifs avec une probabilité assez forte pour suggérer une hypothèse alternative. Ce phénomène est particulièrement vrai sur les mots courts. Lisser le signal ou permettre au BLSTM-CTC de mettre en avant plus de possibilités peut être une solution pour tenter d'améliorer les résultats du système. Néanmoins il ne faut pas tomber dans l'excès inverse, proposer un panel trop large d'hypothèses à chaque trame.

4.3 Conclusion

Ce chapitre présente deux systèmes de reconnaissance de l'écriture manuscrite basés sur des modèles d'attache aux données différents. D'un côté nous avons un modèle d'attache aux données avec une mémoire intégrée capable d'abstraire des informations spatiale et temporelle : le BLSTM-CTC. De l'autre nous avons un modèle d'attache aux données utilisant un large contexte et dépourvu de mémoire sur ce contexte, mais avec une mémoire sur la décision précédente : le CAC. Sur ces deux systèmes nous avons testé quatre jeux de ca-

ractéristiques transformés dans deux représentations, chacune adaptée pour le classifieur dynamique auquel elles sont fournies. Nos résultats montrent qu'à un niveau de complexité équivalent du système, le BLSTM-CTC est de loin bien plus performant que le CAC. Malgré des améliorations apportées au système CAC/MMC dans [20] ce système n'a pas pu combler les écarts de performances avec le système BLSTM-CTC.

Concernant le BLSTM-CTC nous avons pu montrer que les caractéristiques construites par des opérateurs mathématiques (CDP, HOG, ORB) permettent d'apporter une information que le BLSTM-CTC ne peut pas construire directement à partir des pixels. Nous avons aussi montré que ces différentes caractéristiques apportent des informations différentes puisque les systèmes entraînés sur ces caractéristiques ne commettent pas les mêmes erreurs. Dans le chapitre suivant nous proposons des solutions permettant de combiner le contenu informationnel apporté par ces caractéristiques dans un BLSTM-CTC afin d'améliorer les résultats. Pour cela nous explorons plusieurs stratégies de combinaisons, en explorant différents niveaux et manières de combiner des informations au travers du BSLTM-CTC.

Chapitre 5

Exploration des stratégies de combinaisons de réseaux récurrents

Le chapitre précédent présentait une comparaison de systèmes de reconnaissance de l'écriture manuscrite, qui différaient d'abord au niveau du modèle d'attache aux données et ensuite au niveau de la représentation de l'information utilisée. Nous avons présenté un système Modèle de Markov Cachés, un système hybride réseau de neurones/MMC, un système hybride « Bidirectional Long Short Term Memory - Connectionist Temporal Classification » (BLSTM-CTC) / MMC, et un système hybride Champs Aléatoires Conditionnels (CAC) / MMC.

Le modèle neuro-markovien BLSTM-CTC/MMC s'est révélé bien plus performant. Cette différence peut s'expliquer par trois différences majeures entre les systèmes. Premièrement, le système BLSTM-CTC/MMC est capable de générer de lui même une segmentation durant l'apprentissage du modèle d'attache aux données, il ne requiert pas d'étiquetage préalable de la base d'apprentissage. Par conséquent le système n'est donc pas biaisé par d'éventuels mauvais choix d'un étiquetage initial, contrairement au système CAC/MMC et au réseau de neurones du système réseau de neurones/MMC qui eux requièrent cet étiquetage préalable. Deuxièmement, la mémoire du BLSTM, c'est à dire les unités LSTM, synthétisent mieux les informations importantes au niveau temporel et spatial que l'utilisation du contexte sur les observations du CAC qui est sans doute trop large et ne permet pas d'extraire les informations pertinentes, ni de suivre leur évolution au cours du temps. Troisièmement, nous avons constaté des problèmes de transitions dans le système CAC dans le cadre de son utilisation pour la reconnaissance de l'écriture. Les transitions des sorties semblent biaisées vers la transition intra-caractère, dès qu'il détecte un caractère le CAC va donc rester sur ce caractère sur un trop grand nombre de trames. On risque donc d'omettre des caractères durant la reconnaissance.

Ceci est lié au fait que le CAC ne prend pas les dépendances à long terme. L'utilisation de la sortie à l'instant $t-1$ pour déterminer la sortie à t par le CAC, l'utilité de cette information semble être au détriment des résultats. Nous avons proposé en conclusion des améliorations pour ces deux systèmes, et dans ce chapitre nous nous intéressons en particulier au système BLSTM-CTC/MMC.

Nos travaux sur le système BLSTM-CTC ont montré des performances très intéressantes sur la tâche de reconnaissance. Dans le cadre de ce système nous avons intégré et comparé différentes caractéristiques fournies à ce système, dans le but de déterminer les caractéristiques les plus adaptées. Les quatre caractéristiques comparées extraient différentes informations d'une image, via des opérateurs mathématiques différents. L'intensité lumineuse des pixels est la représentation de l'information la plus élémentaire pour une image, c'est une caractéristique de bas-niveau. Les Caractéristiques Des Pixels (CDP) [82] résument l'information contenue dans les pixels tout en spécialisant cette information à la reconnaissance de l'écriture, il s'agit là aussi d'une caractéristique de bas niveau. Les Histogrammes de Gradients Orientés (« Histograms of Oriented Gradients », HoG) [48] offrent un niveau d'abstraction supplémentaire par rapport aux deux caractéristiques précédentes en utilisant l'information des contours de l'image, il s'agit cette fois d'une caractéristique de moyen niveau. « Oriented FAST and Rotated BRIEF » (ORB) [37] offre comme HoG un niveau d'abstraction supplémentaire et utilise des comparaisons d'intensités lumineuses à plusieurs échelles et orientations pour produire un vecteur de caractéristiques de grande dimension. Nous avons constaté que les différentes caractéristiques conduisent les systèmes à effectuer des erreurs différentes. Les erreurs étant différentes et complémentaires, il est alors possible d'imaginer une architecture permettant la correction des erreurs par la combinaison des informations des différentes caractéristiques, ou des représentations internes, ou les sorties du BLSTM-CTC.

Dans ce chapitre nous présentons nos contributions au sujet des stratégies de combinaisons des représentations dans le système BLSTM-CTC/MMC. Nous nous intéressons en particuliers à la combinaison au niveau du modèle d'attache aux données, c'est à dire le BLSTM-CTC. La combinaison des sorties au niveau du modèle d'attache aux données a déjà été traitée dans différentes publications [4, 5, 18] qui présentent des stratégies de combinaison au niveau mot (moyenne, vote majoritaire, combinaison par des algorithmes de classification) elle ne sera donc pas abordée dans nos travaux. Le contenu de ce chapitre

a fait l'objet de publications dans les conférences ICPRAM et DRR [129, 130]. Nous présentons dans un premier temps l'ensemble des combinaisons étudiées, puis les résultats obtenus.

5.1 Combinaisons de BLSTM-CTC

Nous avons constaté que pour un même système des caractéristiques différentes conduisent les BLSTM-CTC à effectuer des erreurs différentes. Le tableau 4.4 présente différentes erreurs du BLSTM-CTC et le tableau 4.3 présente les distances de Levenshtein pour le Top 1 des différents systèmes. On observe que certains BLSTM-CTC se trompent sur certaines images tandis que d'autres proposent des sorties correctes, dans d'autres cas aucun BLSTM-CTC ne permet d'obtenir la sortie désirée mais des caractères de la solution recherchée apparaissent dans les sorties proposées.. Par exemple dans le cas de l'image du mot « société », ORB obtient correctement le « s » et HoG obtient correctement le « té » final. À partir de ces exemples nous pouvons voir qu'une combinaison des BLSTM-CTC pourrait permettre d'améliorer les sorties.

Certaines caractéristiques utilisées sont plus adaptées à certains styles d'écritures ou à certains caractères. La combinaison des informations est généralement reconnue comme permettant d'améliorer les systèmes complémentaires [18, 33, 73, 194], cependant il n'y a pas de consensus sur le niveau auquel il est nécessaire de réaliser la fusion. Par niveau nous entendons différentes parties de l'architecture du BLSTM-CTC : en entrée du système (bas niveau), dans le système (moyen niveau), ou en sortie du système (haut niveau). Le BLSTM-CTC dispose de ces trois niveaux pour réaliser des opérations de combinaison : le bas niveau correspond à une intégration en amont des représentations de l'information, le niveau intermédiaire correspond à une combinaison des représentations intermédiaire fournies par les BLSTM et avant le CTC, et le haut niveau correspond à une fusion tardive des sorties de chaque BLSTM-CTC.

Notre contribution est centrée sur l'exploration de différentes stratégies pour combiner des systèmes BLSTM-CTC, ce qui à notre connaissance n'a jamais été étudiée dans la littérature. Afin de réaliser ces combinaisons nous nous basons sur les caractéristiques et le système BLSTM-CTC/MMC présentés dans le chapitre précédent en section 4.1. Nous détaillons d'abord l'ensemble des stratégies de combinaisons explorées, puis nous présentons les résultats obtenus.

5.1.1 Combinaison bas niveau

La combinaison de bas niveau est une combinaison relativement bien connue dans la littérature [33, 105, 107]. Cette stratégie de combinaison permet de mesurer la capacité du BLSTM-CTC à construire une représentation interne efficace pour représenter les particularités spatiales et temporelles du signal qu'est une image de mot. Cette combinaison s'effectue par une fusion des caractéristiques et nécessite l'apprentissage d'un unique BLSTM-CTC. Cette fusion est une opération de concaténation des vecteurs, voir figure 5.1. Nous considérons un ensemble E composés de F familles de caractéristiques $E = \{E_1, E_2, \dots E_F\}$. Soit une famille de caractéristiques $E_f \in E$, une séquence d'entrée $x^{E_f} = (x_1^{E_f}, x_2^{E_f}, \dots x_T^{E_f})$. Le vecteur de caractéristiques concaténées $x_t^{E_C}$ à l'instant t est défini comme :

$$x_t^{E_C} = \begin{pmatrix} x_t^{E_1} \\ x_t^{E_2} \\ \dots \\ x_t^{E_F} \end{pmatrix} \tag{5.1}$$

et la nouvelle séquence d'entrée $x^{E_C} = (x_1^{E_C}, x_2^{E_C}, \dots x_T^{E_C})$. En fournissant une variété de caractéristiques, travaillant sur des échelles et une abstraction des données différentes (caractéristiques de bas, moyen et haut niveau, voir section 3.4) nous pouvons mesurer l'apport de chaque caractéristique en examinant l'évolution des performances du système.

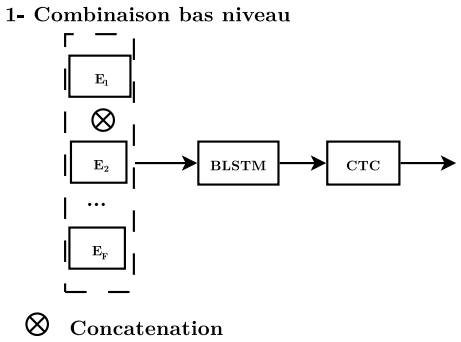


FIGURE 5.1 – Combinaison bas niveau

Après avoir présenté la combinaison bas-niveau, nous présentons maintenant la combinaison intermédiaire qui combine une abstraction des caractéristiques fournies par les sorties du BLSTM.

5.1.2 Combinaison au niveau intermédiaire

La combinaison intermédiaire s'intéresse à la représentation en sortie des BLSTM, en particulier à l'abstraction qu'ils produisent à partir des caractéristiques d'entrée. Cette combinaison est inspirée des méthodes d'apprentissage des réseaux de neurones profonds [12]. Cette méthode combine plusieurs BLSTM à l'aide d'un réseau de combinaison.

Un ensemble de F réseaux de neurones BLSTM-CTC est d'abord appris sur l'ensemble E des caractéristiques. De cette manière les BLSTM-CTC sont spécialisés pour une famille de caractéristiques et ont été optimisés pour une sortie mot. Les couches CTC et les poids qui leur sont attachés sont ensuite supprimés. Les BLSTM sont alors considérés comme des extracteurs de caractéristiques de haut niveau, de manière similaire à une couche encodeuse dans un réseau profond auto-encodeur, voir section 2.1.2 pour plus de détails.

Les caractéristiques produites par les différents BLSTM sont ensuite concaténées pour former un nouveau vecteur. Soit une famille de caractéristiques $E_f \in E$, une séquence en sortie du BLSTM $h^{E_f} = (h_1^{E_f}, h_2^{E_f}, \dots h_T^{E_f})$. Le vecteur des représentations concaténées des sorties des BLSTM $h_t^{E_C}$ à l'instant t est défini comme :

$$h_t^{E_C} = \begin{pmatrix} h_t^{E_1} \\ h_t^{E_2} \\ \dots \\ h_t^{E_F} \end{pmatrix} \tag{5.2}$$

Ce nouveau vecteur est ensuite utilisé comme séquence d'entrée d'un classifieur. Nous proposons deux stratégies de combinaison des sorties des BLSTM pour le niveau intermédiaire.

Dans le premier cas nous réalisons un apprentissage sans utilisation du contexte. Une couche CTC est apprise de manière indépendante (sans modification des poids des BLSTM précédemment appris) afin de combiner les sorties des BLSTM sur une trame indépendante pour produire les sorties (les probabilités a posteriori des caractères).

Dans le second cas une mémoire est utilisée afin d'exploiter les dépendances de longues portées entre les signaux des sorties des BLSTM. En effet l'ensemble des F BLSTM-CTC a été initialement appris sur des caractéristiques différentes, cela a pour effet d'avoir des sorties différentes sur une même image. C'est à dire que pour une image de mot de longueur t, le signal de sortie de deux BLSTM-CTC sont différents, en particulier les instants où les BLSTM-CTC se prononcent. Pour rappeler le paradoxe de Sayre, les deux

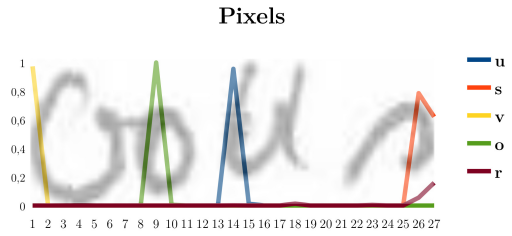
systèmes peuvent reconnaître un même caractère, mais ne le détectent pas nécessairement au même instant : il y a un problème de synchronisation des sorties. Ceci est une particularité de ces systèmes qui ne localisent pas le caractère reconnu du fait de la présence du label joker. La figure 5.2 illustre ce problème, plusieurs BLSTM-CTC sont appris sur des données identiques mais avec des caractéristiques différentes. Le pic représentant la décision *s* est généré sur des trames différentes : durant les trames 26 et 27 dans le cas des pixels, durant la trame 22 pour CDP et ORB, durant la trame 23 pour HoG. Bien que temporellement proche, un CTC seul ne possède pas de mémoire de cette décision, et peut donc être sujet à des erreurs en décision. C'est pour cela que nous désirons mesurer l'impact d'un apprentissage du contexte.

Les stratégies de combinaisons des BLSTM sont donc :

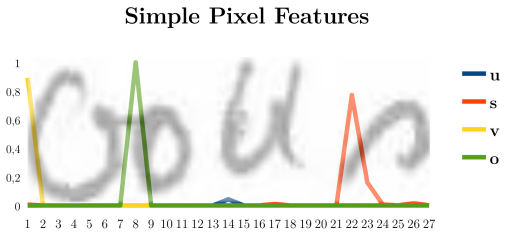
2.1 Combinaison par CTC ;

2.2 Combinaison par BLSTM-CTC.

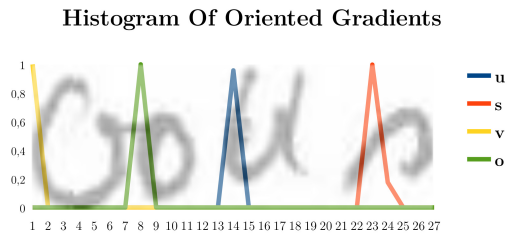
Ces deux combinaisons sont représentées en figure 5.3.



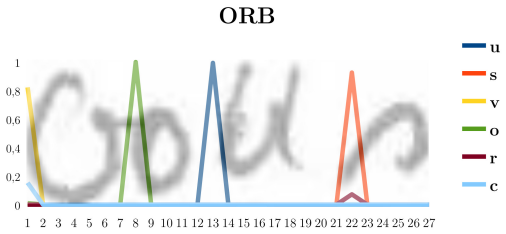
(a) Signal de sortie du CTC utilisant les Pixels



(b) Signal de sortie du CTC utilisant les CDP [81]



(c) Signal de sortie du CTC utilisant les caractéristiques HoG [48]



(d) Signal de sortie du CTC utilisant les caractéristiques ORB [37]

FIGURE 5.2 – Signaux en sortie des BLSTM-CTC appris sur des caractéristiques différentes

Les combinaisons de niveau intermédiaire utilisent l'abstraction des caractéristiques produite par différents BLSTM, chacun spécialisé sur un type

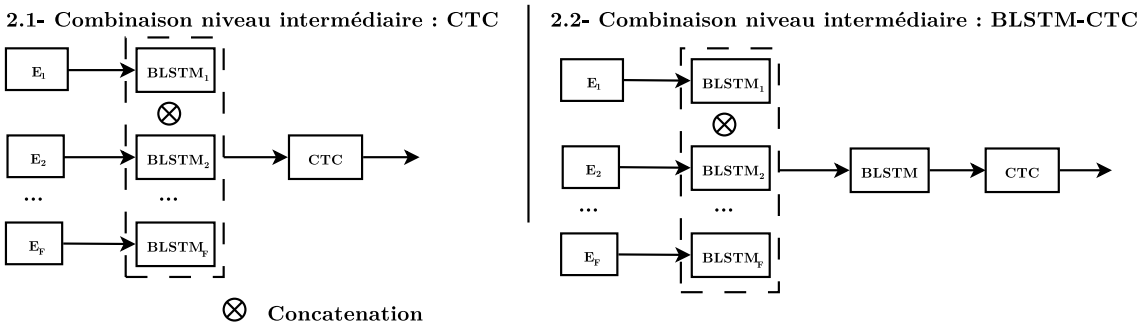


FIGURE 5.3 – Combinaisons niveau intermédiaire

de caractéristiques. De manière similaire nous proposons des combinaisons de haut-niveau, cette fois ci basées sur les sorties de différents BLSTM-CTC, chacun spécialisé sur un type de caractéristiques.

5.1.3 Combinaison de haut-niveau

Les combinaisons de haut niveau sont les combinaisons des sorties des CTC, elles utilisent la spécialisation des réseaux sur un type de caractéristiques et combinent les informations de différents réseaux dans l'objectif d'améliorer la décision.

Un ensemble de F réseaux de neurones BLSTM-CTC est d'abord appris sur l'ensemble E des caractéristiques. Les BLSTM-CTC sont spécialisés pour une caractéristique donnée. Les sorties des CTC sur une même image de mot sont utilisées pour effectuer les nouvelles combinaisons, c'est à dire les probabilités a posteriori des caractères. Nous proposons cinq stratégies de combinaisons différentes :

3.1 Moyenne des sorties ;

3.2 Moyenne des sorties, puis apprentissage d'un BLSTM-CTC ;

3.3 Maximum des sorties ;

3.4 Maximum des sorties, puis apprentissage d'un BLSTM-CTC ;

3.5 Concaténation des sorties, puis apprentissage d'un BLSTM-CTC.

La figure 5.4 représente l'ensemble des combinaisons de haut niveau présentées.

La combinaison par la moyenne [108] a pour objectif de lisser les différentes erreurs commises par les BLSTM-CTC. La moyenne est définie comme la somme pondérée par le nombre de classifieurs BLSTM-CTC à combiner. Soit une caractéristique $E_f \in E$, une séquence de sortie $y^{E_f} = (y_1^{E_f}, y_2^{E_f}, \dots y_T^{E_f})$.

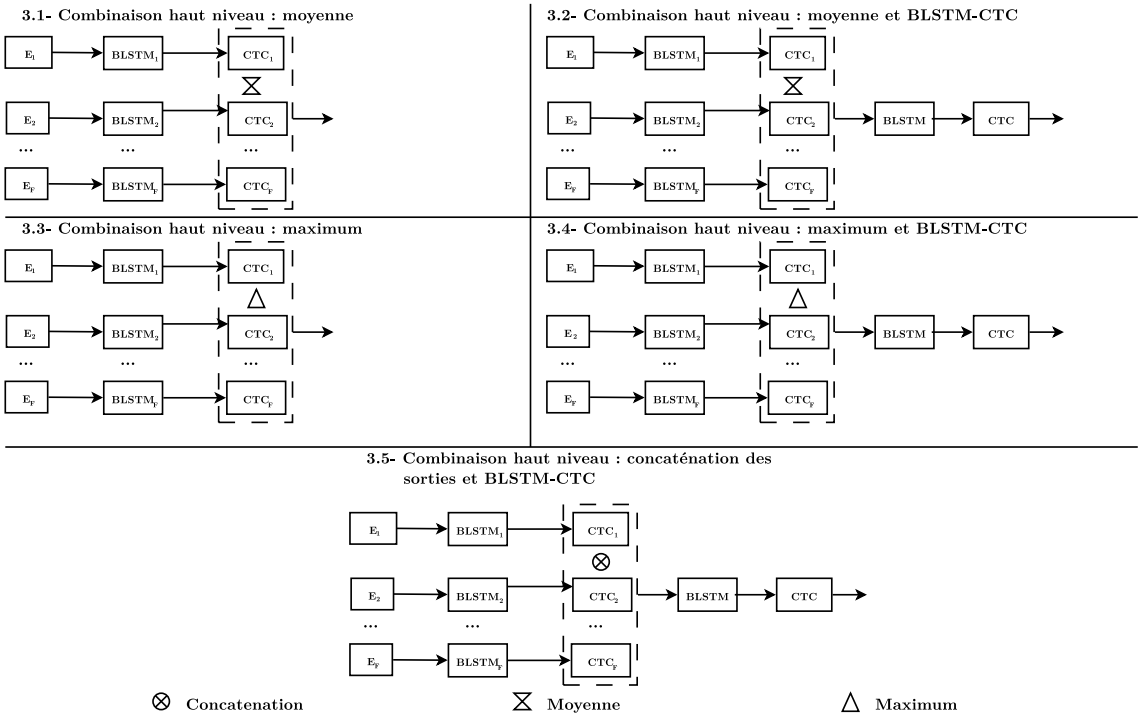


FIGURE 5.4 – Combinaison haut niveau

Le vecteur des sorties concaténées $y_t^{E_C}$, à l'instant t, pour un caractère $l' \in L'$ ou L' est l'ensemble des labels (blanc compris), est défini comme :

$$y_{t,l'}^{E_C} = \frac{y_{t,l'}^{E_1} + y_{t,l'}^{E_2} + \cdots + y_{t,l'}^{E_F}}{F} \tag{5.3}$$

Les sorties sont ensuite normalisées pour obtenir des sorties $y_{t,l'}^{E_C} \in [0;1]$ avec $\sum_{l'}^{L'} y_{t,l'}^{E_C} = 1$, afin d'obtenir des probabilités a posteriori sur les caractères (ces opérations sont similaires à celle d'une couche Softmax voir 2.2.3.6). La combinaison par moyenne des sorties met en exergue des décisions fortes de plusieurs BLSTM-CTC pour un même caractère à un instant donné, tandis qu'elle atténue les signaux forts prononcés uniquement par un BLSTM-CTC.

La combinaison par maximum [103, 108], a pour but de mettre en avant la meilleure sortie proposée parmi l'ensemble des sorties des BLSTM-CTC. Le maximum est défini comme :

$$y_{t,l'}^{E_C} = \max(y_{t,l'}^{E_1}, y_{t,l'}^{E_2}, \ldots, y_{t,l'}^{E_F}) \tag{5.4}$$

Les sorties sont ensuite normalisées comme précédemment. La combinaison par maximum permet de mettre en avant pour chaque caractère une décision forte, quel que soit le CTC qui a décidé du caractère. À la différence de la

moyenne, le maximum ne détruit pas des décisions fortes, même si peu de CTC ont considéré cette solution.

La concaténation a pour but de rassembler les informations et de laisser un classifieur apprendre un schéma de combinaisons des probabilités a posteriori permettant d'améliorer le système. La concaténation des sorties est définie comme la fusion des vecteurs de sorties des CTC à un instant t. Soit une caractéristique $E_f \in E$, une séquence de sortie $y^{E_f} = (y_1^{E_f}, y_2^{E_f}, \dots y_T^{E_f})$. Le vecteur des sorties concaténées $y_t^{E_C}$ à l'instant t est défini comme :

$$y_t^{E_C} = \begin{pmatrix} y_t^{E_1} \\ y_t^{E_2} \\ \dots \\ y_t^{E_F} \end{pmatrix} \tag{5.5}$$

et la nouvelle séquence de sortie $y^{E_C} = (y_1^{E_C}, y_2^{E_C}, \dots y_T^{E_C})$.

La combinaison 3.2 et 3.4 ajoute respectivement à la combinaison 3.1 et 3.3 un BLSTM-CTC. L'objectif est similaire à celui présenté dans la sous-section précédente 5.1.2, c'est à dire apprendre des relations temporelles afin de remédier aux disparités temporelles pour une décision caractère identique. Dans le cas de la combinaison 3.5 cela permet aussi de revenir sur une échelle de dimension utilisable pour un MMC. Les MMC ne supportant que des vecteurs de faible dimensions (~ 100), il n'est pas concevable de leur donner un vecteur contenant $|L'| \times R >> 100$. Pour la combinaison 3.5 nous apprenons donc un BLSTM-CTC à la fois pour une utilisation du contexte et à la fois pour réduire les dimensions.

Après avoir présenté l'ensemble des combinaisons étudiées, nous présentons maintenons les résultats obtenus sur la base RIMES.

5.2 Expériences

Dans cette section nous présentons les expériences produites et les résultats obtenus dans le cas de la tâche WR2 de la compétition ICDAR 2009 [85], qui évalue notre système sur une tâche de reconnaissance de mots isolés. Dans le but d'éviter et de mesurer un éventuel sur-apprentissage sur cette base nous réalisons un nouveau découpage de la base d'apprentissage, ce qui nous mène à avoir deux conditions différentes d'apprentissage. Nous présentons nos deux conditions expérimentales puis les résultats obtenus.

5.2.0.1 Bases

Réaliser un apprentissage sur l'ensemble de la base RIMES pour les stratégies de combinaisons 2.1, 2.2 ,3.2, 3.4 et 3.5 nous mène à utiliser deux fois l'ensemble de la base : une fois pour le BLSTM-CTC initial et la seconde pour le CTC ou le BLSTM-CTC suivant. Afin d'éviter un éventuel sur-apprentissage et de le mesurer, nous effectuons deux types d'apprentissage différents : avec le découpage standard de RIMES (A1), avec des sous-bases générées depuis la base RIMES(A2).

Dans A1 tous les classifieurs nécessitant un apprentissage sont appris sur l'ensemble de la base d'apprentissage de RIMES. Ils sont ensuite validés et testés sur les bases prévues.

Dans A2 afin d'éviter un phénomène de sur-apprentissage pour les combinaisons intermédiaire et les combinaisons de haut niveau, nous proposons un nouveau partitionnement de la base pour l'apprentissage des BLSTM-CTC des caractéristiques.

La base d'apprentissage et de validation sont combinées. Cette nouvelle base est ensuite divisée en F sous bases équilibrées, où F est le nombre de familles de caractéristiques utilisées. Chaque sous base est ensuite divisée suivant un ratio $80\% - 20\%$ avec respectivement la sous base d'apprentissage et la sous base de validation. Les classifieurs des caractéristiques sont appris sur ces sous bases. Dans le cas de la combinaison bas niveau, l'ensemble de la base est utilisé pour apprendre le BLSTM-CTC. Dans le but d'effectuer une comparaison équitable des combinaisons 2.1, 2.2, 3.2, 3.4, 3.5, nous utilisons les classifieurs appris sur les sous bases puis nous utilisons l'ensemble de la base d'apprentissage pour apprendre les pondérations du CTC ou du BLSTM-CTC. Ainsi toutes les combinaisons apprennent sur la base d'apprentissage complète et non uniquement sur des sous bases. Ceci permet de limiter le sur-apprentissage global.

5.2.0.2 Résultats

Les résultats sont présentés en combinant les caractéristiques suivantes : pixels de l'image (section 4.1.3.1), les Caractéristiques des Pixels (section 4.1.3.2), Histogramme de Gradients Orientés (section 4.1.3.3) et ORB (section 4.1.3.4).

Dans l'objectif de mesurer l'influence du nombre de caractéristiques dans la combinaison et l'augmentation de la taille du vecteur (dans le cas du système 1), nous étudions plusieurs combinaisons de caractéristiques :

1. Deux caractéristiques : CDP et HoG dans les conditions A1 ;
2. Trois caractéristiques : pixels, CDP et HoG dans les conditions A1 ;
3. Quatre caractéristiques : pixels, CDP, HoG, ORB dans les conditions A1 et dans les conditions A2.

L'ensemble des résultats est compilé dans le tableau 5.1.

Les tableaux présentent dans un premier temps les résultats des caractéristiques seules, puis les résultats des différentes combinaisons. Nous comparons ces combinaisons au travers de l'erreur mot (« Word Error Rate ») brut, c'est à dire la sortie avant le décodage dirigé par le lexique effectué par le MMC, ainsi qu'au travers du Top 1 et du Top 10 obtenus par le décodage dirigé par le lexique du MMC.

Dans un premier temps nous constatons l'importance du décodage dirigé par le lexique. Les systèmes gagnent entre 50% et 15% de bonne reconnaissance entre le WER brut et le Top 1. L'utilisation d'un décodage par le lexique est donc très important pour la reconnaissance de l'écriture.

Nous constatons que la combinaison bas-niveau (système 1), c'est à dire la concaténation des caractéristiques, est la plus performante sur l'ensemble des résultats à l'exception de la combinaison à quatre caractéristiques dans les conditions A1. Les résultats positifs de ces expériences montrent la performance du réseau BLSTM-CTC à modéliser de manière efficace le contexte temporel et spatial. Cependant ce système est celui qui évolue le moins lorsqu'on ajoute des caractéristiques, bien que les réseaux BLSTM-CTC aient évolués en terme de quantité de neurones de la couche cachée pour prendre en compte l'augmentation des dimensions en entrée. Il n'y a pas d'améliorations constatées sur le Top 1 entre la combinaison à deux, trois ou quatre caractéristiques dans les conditions A1 et le recouvrement des solutions en top 1 est très similaire. Ce phénomène peut être imputé à la similarité du contenu informationnel entre les caractéristiques pixels et CDP dans le cas de la non-évolution entre la combinaison à deux ou trois caractéristiques. Néanmoins après avoir ajouté la caractéristique ORB aucune amélioration n'est constatée, il y a donc une saturation de la capacité de l'apprentissage d'un unique BLSTM-CTC pour apprendre un vaste nombre de caractéristiques.

À l'inverse les combinaisons de haut niveau présentent les améliorations des performances les plus importantes. Cependant dans la combinaison à deux caractéristiques dans les conditions A1 seul le maximum (système 3.3) offre des solutions intéressantes dans le Top 1. Nous expliquons les résultats peu intéressants de la moyenne (système 3.1) et de la moyenne et BLSTM-CTC

(système 3.2) dans le cas à deux caractéristiques dans les conditions A1, simplement par le fait qu'il n'y a pas assez de caractéristiques pour produire des valeurs cohérentes. En effet si à un instant t les deux BLSTM-CTC ont des sorties différentes avec des valeurs à 1 (un pic de sortie) on se retrouve alors dans un cas d'équiprobabilités entre des caractères. Le MMC doit donc utiliser des équiprobabilités pour déterminer le bon mot, or dans le cas des mots courts « de », « du » ou « le », « la », « lu », cette équiprobabilité va être néfaste à la décision mot du MMC. C'est ce que nous constatons en observant les résultats produits. En revanche avec trois ou quatre caractéristiques la moyenne (système 3.1) présente des résultats plus intéressants que les autres combinaisons haut-niveau. Avec plus de caractéristiques, la moyenne effectue un lissage plus pertinent des sorties des différents BLSTM-CTC. Il est intéressant de constater que la moyenne produit de meilleurs résultats en Top 10 que la combinaison bas-niveau (système 1), les hypothèses de caractères produites par la moyenne sont donc plus pertinentes.

La moyenne et BLSTM-CTC (système 3.2) montre la même tendance que la moyenne sur le Top 10 et est aussi de 6% le système le plus performant sur le WER brut pour quatre caractéristiques. Ces performances importantes sur le WER brut sont intéressantes pour des systèmes où un décodage dirigé par le lexique peut ne pas être applicable (comme pour des entités nommées tels des références produits, ou des noms de famille, ou des séquences numériques). Un autre phénomène intéressant est la stabilité des résultats entre les deux conditions d'apprentissage A1 et A2, avec une légère perte de performance au niveau du WER brut pour la condition A2.

Le maximum (système 3.3) et le maximum et BLSTM-CTC n'apportent pas d'améliorations notables par rapport aux systèmes décrits précédemment.

Il est intéressant et étonnant de constater que la concaténation des sorties et BLSTM-CTC (système 3.5) ne donne pas la meilleure des performances. Cela signifie que le BLSTM-CTC effectuant la réduction de dimension ne parvient pas à effectuer une opération de moyenne ou de maximum via ses unités neuronales.

Concernant les systèmes avec des combinaisons intermédiaires (système 2.1 et 2.2) nous constatons que ces systèmes sont généralement compétitifs avec la combinaison bas niveau dans les conditions A1 sur le Top 1. Ils deviennent plus performant avec un nombre plus important de caractéristiques. En revanche dans les conditions A2 il y a une perte de 1% à 2% dans le Top 1 Dans le cas d'un sur-apprentissage pour un système, la combinaison au niveau

intermédiaire permet d'optimiser séparément les réseaux de neurones sur une famille de caractéristiques, ce qui leur permet d'obtenir des représentations des données intéressantes au niveau du BLSTM. Cette représentation est utilisée pour apprendre les pondérations d'un CTC (système 2.1) ou un BLSTM-CTC (système 2.2), cet apprentissage est très similaire à celui d'un réseau de neurones profond [12, 176], et permet d'utiliser des caractéristiques possédant un fort niveau d'abstraction. Avec une multitude de caractéristiques il semble plus pertinent d'utiliser cette solution, en particulier avec des caractéristiques de très hautes dimensions (512 valeurs pour ORB).

Le tableau 5.2 référence l'évolution des résultats du tableau 4.4 obtenus dans le cas des combinaisons 1, 2.1 et 3.2. Nous constatons les améliorations apportées par les stratégies de combinaisons sur ces différentes images. Certaines images sont encore problématiques, mais elles restent aussi difficiles à lire pour un humain, en particulier l'image du mot « prend ».

En comparant les résultats pour les combinaisons avec quatre caractéristiques dans les conditions A1 et A2 nous constatons des résultats intéressants au niveau de l'amélioration des performances. En effet les performances des systèmes initiaux dans les conditions A2 sont bien inférieures à celle dans les conditions A1, cela s'explique simplement par une différence de l'échantillon utilisé pour l'apprentissage, néanmoins les combinaisons au niveau intermédiaire et au haut niveau sont très similaires au niveau du Top 1 et Top 10.

5.2.1 Conclusion

Les stratégies de combinaisons que nous avons mises en place pour un réseau récurrent de type BLSTM-CTC ont la capacité de transformer une image de mot en une séquence de caractères numériques. Ces combinaisons offrent de meilleures performances que des BLSTM-CTC appris sur des caractéristiques prises isolément. Les stratégies de combinaisons proposées offrent différents avantages, trois d'entre elles sont particulièrement intéressantes pour leur stabilité ou les performances apportées. La combinaison bas-niveau (système 1) offre une amélioration des performances dans le cas où l'on combine deux à trois caractéristiques de faibles dimensions. Néanmoins nos résultats montrent que ce système stagne rapidement avec une augmentation de la taille du vecteur d'entrée et du nombre de neurones dans la couche cachée. L'apprentissage d'un système aussi complexe ne permet pas d'abstraire autant d'informations spatiales et temporelles depuis des vecteurs de très grandes dimensions. Au contraire les systèmes 2.1 et 3.2 divisent l'abstraction des informations spatiales

et temporelles entre plusieurs BLSTM ou BLSTM-CTC, puis rassemblent les informations obtenues afin d'améliorer la reconnaissance caractère. Cette division permet à un BLSTM ou BLSTM-CTC de se spécialiser dans la reconnaissance et l'abstraction de certaines informations contenues dans les trames tout en évitant le bruit apporté par un vecteur de trop grande dimension.

En comparant les résultats obtenus dans les conditions A1 et A2 nous pouvons voir que le sur-apprentissage ne semble pas significatif, malgré la taille réduite de la base RIMES. Malgré cela nous ne soutenons pas une systématisation d'une stratégie d'apprentissage qui réutilise de multiples fois la même base pour apprendre des systèmes en cascade, ce résultat est grandement dépendant de la base. En effet la base RIMES malgré sa faible taille reste très diverse dans les styles d'écriture et le nombre de mots différents. Dans le cas d'une base moins diverse ou avec moins d'échantillons il est préférable d'utiliser une stratégie d'apprentissage en subdivisant la base d'apprentissage.

Finalement si nous comparons nos résultats à ceux de la campagne ICDAR 2009 [85], voir le tableau 5.3, nous constatons que nous ne sommes pas au dessus du système MDRNN-CTC, mais nous en sommes très proches cependant. Nos combinaisons offrent un temps d'apprentissage plus court que le MDRNN-CTC et ont une convergence plus facile à maîtriser.

Les hyper-paramètres de nos réseaux BLSTM-CTC (nombre de couches, nombre de neurones, pondération des erreurs, etc.) et les paramètres du système (longueur des trames, taille des sous fenêtres ORB et HoG, etc.) n'ont pas tous été explorés, les apprentissages étant relativement longs et le hasard de l'initialisation des réseaux empêche une recherche exhaustive menant à la solution optimale. Ce problème des réseaux de neurones est bien connu [14]. Nous avons donc préféré obtenir des réseaux « moyens » présentant des performances non-optimales mais néanmoins correctes. Nous avons pu montrer que via ces réseaux sous-optimaux nous nous approchons des meilleurs résultats d'ICDAR 2009, ce qui est intéressant pour construire des systèmes complexes sur de grandes bases. Dans le cas d'un système appris sur une base très large, nous pouvons subdiviser la base et déléguer l'apprentissage des sous-bases à des couples caractéristiques–BLSTM-CTC, nous réduisons donc les apprentissages à réaliser avant les combinaisons tout en minimisant les pertes de performances.

5.3 Conclusion

Dans ce chapitre nous avons mis en place un ensemble de stratégies de combinaisons de réseaux récurrents BLSTM-CTC pour la reconnaissance de l'écriture dans le cas de mots isolés. Les différentes stratégies de combinaisons sont construites pour chacun des niveaux de la structure d'un BLSTM-CTC, le bas niveau intègre les caractéristiques en un unique vecteur traité par un unique BLSTM-CTC, tandis que les combinaisons intermédiaires délèguent le traitement des caractéristiques à plusieurs BLSTM et un système final prend la décision. De manière similaire les combinaisons haut-niveau subdivisent le traitement des caractéristiques entre plusieurs BLSTM-CTC avant de fusionner les systèmes via des opérateurs de combinaison ou des classifieurs.

Nous avons montré l'efficacité de la combinaison bas niveau (intégration en amont des caractéristiques) pour l'intégration d'un faible nombre de caractéristiques de petites dimensions, néanmoins cette combinaison stagne au niveau des performances. La combinaison intermédiaire avec un CTC et la combinaison de haut-niveau avec moyenne et BLSTM-CTC sont les plus intéressantes avec un grand nombre de caractéristiques, et en particulier avec des caractéristiques de grandes dimensions.

La reconnaissance de l'écriture n'est pas une fin en soi d'un point de vue industriel, il ne s'agit que d'un moyen pour automatiser un système ou bien apporter une aide à l'utilisateur. La reconnaissance de l'écriture manuscrite peut par exemple servir à l'indexation de documents, afin d'effectuer par la suite des requêtes texte, permettant ainsi à un utilisateur de trouver un document image possédant un contenu d'intérêt. Dans le chapitre suivant nous présentons deux applications à nos systèmes de reconnaissance de l'écriture manuscrite.

Système	Conditions	WER brut	Top 1	Top 10
Sans combinaison				
Pixels	A1	41.95	13.19	3.97
Pixels	A2	52.22	19.27	8.48
CDP	A1	42.23	13.06	4.34
CDP	A2	56.66	21.6	9.37
HOG	A1	37.62	12.49	4.47
HOG	A2	51.15	17.76	6.21
Orb	A1	34.54	10.67	3.78
Orb	A2	51.95	18.84	6.79
1-Combinaisons bas-niveau				
CDP et HoG	A1	31.87	9.52	3.16
Pixels, CDP, HoG	A1	32.04	9.53	3.01
Pixels, CDP, HoG et Orb	A1 et A2	31.2	9.5	3.15
2-Combinaisons niveau intermédiaire				
2.1-Concaténation des sorties et CTC :				
CDP et HoG	A1	34.6	11.31	3.92
Pixels, CDP, HoG	A1	33.16	10.06	3.11
Pixels, CDP, HoG et Orb	A1	28.03	**8.1**	**2.01**
Pixels, CDP, HoG et Orb	A2	37.48	11.64	4.1
2.2-Concaténation des sorties et BLSTM-CTC :				
CDP et HoG	A1	35.7	9.52	3.73
Pixels, CDP, HoG	A1	32.89	9.6	2.29
Pixels, CDP, HoG et Orb	A1	27.84	8.92	2.54
Pixels, CDP, HoG et Orb	A2	35.76	10.75	3.44
3-Combinaisons haut-niveau				
3.1-Moyenne :				
CDP et HoG	A1	92.8	40.62	20.31
Pixels, CDP, HoG	A1	56.8	10.92	2.35
Pixels, CDP, HoG et Orb	A1	66.54	10.21	2.97
Pixels, CDP, HoG et Orb	A2	91.51	15.45	4.61
3.2-Moyenne et BLSTM-CTC :				
CDP et HoG	A1	38.5	20.62	5.44
Pixels, CDP et HoG	A1	34.21	13.36	5.68
Pixels, CDP, HoG et Orb	A1	**24.24**	8.94	2.78
Pixels, CDP, HoG et Orb	A2	27.7	8.94	2.78
3.3-Maximum :				
CDP et HoG	A1	59.4	10.6	2.25
Pixels, CDP et HoG	A1	74.44	12.09	2.93
Pixels, CDP, HoG et Orb	A1	74.66	12.18	2.35
Pixels, CDP, HoG et Orb	A2	95.5	17.67	5.28
3.4-Maximum et BLSTM-CTC :				
CDP et HoG	A1	34.64	13.4	5.58
Pixels, CDP et HoG	A1	34.25	12.45	4.19
Pixels, CDP, HoG et Orb	A1	37.84	9.51	3.08
Pixels, CDP, HoG et Orb	A2	30.61	13.05	4.48
3.5-Concaténation des sorties et BLSTM-CTC :				
CDP et HoG	A1	42.0	23.99	7.43
Pixels, CDP etHoG	A1	35.73	15.53	6.36
Pixels, CDP, HoG et Orb	A1	30.34	10.67	3.78
Pixels, CDP, HoG et Orb	A2	30.99	11.64	4.48

TABLE 5.1 – Résultats des différentes stratégies de combinaisons sur la tâche WR2 de la compétition ICDAR 2009

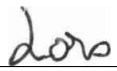
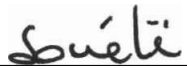
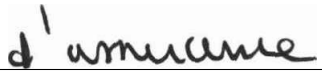
Image	Système 1	Système 2.1	Système 3.2	Vérité terrain
	Lors	dans	Lors	Lors
	société	doute	société	société
	d'assurance	d'assurance	d'épargne	d'assurance
	perdu	prix	prêt	prend
	premier	premier	première	première

TABLE 5.2 – Comparaisons de différentes sorties erronées du Top 1 des stratégies de combinaisons 1, 2.1 et 3.2

Système	Top 1	Top 10
MDRNN-CTC (TUM)	6.83	1.05
Système 2.1	8.1	2.01
Système 3.2	8.94	2.78
Système neuro-markovien (UPV)	13.89	2.05

TABLE 5.3 – Nos résultats comparés à ceux de la compétition ICDAR 2009 [85]

Chapitre 6

Applications

Après avoir présenté une comparaison de différents systèmes de reconnaissance de l'écriture et une contribution permettant d'améliorer les systèmes de reconnaissance en combinant des systèmes BLSTM-CTC, nous présentons dans ce chapitre différentes applications originales de nos contributions. Les applications que nous considérons sont d'une part la reconnaissance de la langue dans des images de textes manuscrits, et d'autre part la détection de mots clefs dans des images de textes manuscrits. Ces applications sont originales pour la reconnaissance de l'écriture, en effet les systèmes de la littérature abordent avec des approches différentes ces problèmes : ils classifient des caractéristiques extraites de l'image sans faire appel à une étape de reconnaissance de l'écriture. Ces approches par classification de caractéristiques sont utilisées car jusqu'à présent les classifieurs dynamiques n'atteignaient pas des performances suffisantes. Cependant elles ne nous paraissent pas nécessairement pertinentes étant donnés les problèmes traités. Nous présentons donc des systèmes exploitant les informations d'un premier étage de reconnaissance de l'écriture pour réaliser ces différentes tâches. Nous présentons dans un premier temps notre système destiné à la reconnaissance de la langue dans une image de texte, puis nous présentons notre système de détection de mots clefs.

6.1 Reconnaissance de la langue

Dans cette section nous présentons une application du système présenté dans le chapitre 4 à la reconnaissance de la langue (ou identification de la langue). La reconnaissance de la langue a pour objectif de distinguer une langue d'une autre [59] dans un document numérique (texte, image, audio). Cette reconnaissance automatique de la langue dans des documents numériques est devenue importante dans un contexte de mondialisation des communications. Même si l'ensemble de la planète utilise 1% des langues pour 99% des communications, on dénombre pas moins de 6000 langues différentes dans le monde, et parmi les dix langues avec le plus d'influence [187] (anglais, français, espagnol,

russe, arabe, chinois, allemand, japonais, portugais, hindi/ourdou) plusieurs ont des racines linguistiques communes (langues latines, langues germaniques) ou partagent des alphabets (ensemble de caractères permettant d'écrire une langue) ou des scripts (style d'alphabet) communs pour l'écriture. Reconnaître la langue d'un document permet par la suite d'appliquer un système de reconnaissance adapté à la reconnaissance (parole, écriture) de cette langue, par exemple dans le cas d'un appel téléphonique [197] sélectionner le modèle de langage adapté à un locuteur français ou anglais pour effectuer la transcription de la parole. La reconnaissance de la langue est également requise lorsque l'on désire traduire un texte depuis une langue vers une langue compréhensible pour un utilisateur, par exemple la traduction automatique d'une page web bengali vers français.

Dans le cas de la reconnaissance de l'écriture, reconnaître une langue permet de choisir le bon système pour la reconnaissance, c'est à dire :

— les bons modèles de caractères,
— le bon lexique,
— le bon modèle de langage.

La manière d'identifier une langue (ou un script) dans un document numérique dépend de la source du document. Un document produit sur machine contiendra les informations de la langue et du script dans l'encodage utilisé (ASCII, UTF-8, UTF-16). Une image de document manuscrit ne contient pas cette information. Il existe cependant différents méthodes permettant de déterminer la langue ou le script de ces documents, beaucoup des travaux réalisés concernent la reconnaissance de la langue dans documents dactylographiés, très peu traitent les documents manuscrits. L'identification de la langue dans des images de documents dactylographiés passe de manière traditionnelle par l'extraction de caractéristiques sur une zone de l'image, puis par l'utilisation de classifieurs pour déterminer la langue contenue dans cette image [114, 117, 168]. Dans les travaux de A.L. Spitz la reconnaissance de la langue est effectuée en deux étapes. Dans un premier temps le script est identifié, dans un second temps la langue est identifiée. Ces travaux traitent deux scripts différents, les scripts Han (3 langues : chinois, coréen et japonais) et le script latin (23 langues : français, anglais, allemand, etc.). La reconnaissance des langues appartenant au script latin est basée sur la classification de caractéristiques provenant du code de contour des caractères. La reconnaissance des langues appartenant au script Han utilise la densité des pixels dans les images de texte afin de classifier une image comme provenant d'une langue ou d'une autre. S.

Lu et C.L. Tan [120] utilisent la longueur des composantes verticales du texte dactylographié comme caractéristiques pour identifier le script, puis utilisent le code de contour des caractères pour identifier la langue. Sur une tâche de reconnaissance de langue sur des documents avec de l'écriture dactylographiée, ils obtiennent un score de reconnaissance correct de 96.88% avec un système traitant 82 langues dans 64 scripts différents. Sur une tâche similaire, mais avec seulement trois langues (anglais, chinois, japonais), Y. Liu, C.C Lin et F. Chang [117] obtiennent un score de 99.83% de bonne reconnaissance avec un système basé sur des caractéristiques de haut niveau (présence de boucles dans les caractères) et de moyen niveau (gradient verticaux à horizontaux). Sur des documents manuscrits J. Hochberg, K. Bowers, M. Cannon, P. Kelly [93] utilisent des informations provenant des composantes connexes (moyenne, écart-type, inclinaison). Sur une base de 500 documents dans six scripts, huit langues produite par 270 scripteurs, ils obtiennent un score de bonne reconnaissance de 88% et plus particulièrement de 85% sur les langues latines (anglais et allemand). Bien que les caractéristiques diffèrent entre les publications [1, 121], la méthode globale pour identifier est la même : extraire des caractéristiques sur l'apparence du texte, appliquer un algorithme de classification pour déterminer la langue. Lorsqu'il s'agit de langues appartenant à différents alphabets ou scripts l'identification est plus facile car la morphologie de l'écriture est très différente. Cependant lorsqu'il s'agit de deux langues partageant le même script la différenciation de ces langues est plus difficile car les caractéristiques sont similaires. Les résultats de ces méthodes sont excellents sur des documents ne contenant que de l'écriture dactylographiée, néanmoins aucun de ces systèmes n'est appliqué sur du manuscrit. La variabilité de l'écriture manuscrite ne permet pas d'avoir des caractéristiques sur l'apparence graphique des caractères qui soient stables et donc ces systèmes ne peuvent être appliqués avec les mêmes taux de réussite.

Dans cette section nous proposons une méthode différente pour identifier des langues appartenant à un même script dans des documents manuscrits. Nous considérons la reconnaissance de script comme un problème résolu [172], et nous nous attachons donc à la séparation des langues dans un script. La méthode que nous proposons est basée sur l'étude statistique d'une langue, c'est à dire des méthodes appartenant au Traitement Automatique du Langage (TAL). L'idée générale est d'effectuer une reconnaissance de l'écriture sans modèle de langage associé sur une image, puis d'utiliser sur la sortie texte une méthode de classification de la langue, qui est basée sur les statistiques

d'apparitions de certains N-grammes de caractères.Nous présentons dans un premier temps notre système de reconnaissance de la langue, puis nous présentons les expériences et les résultats obtenus.

6.1.1 Système de reconnaissance de la langue

Notre système de reconnaissance de la langue a pour but de prendre en entrée une image de texte et de donner en sortie la langue dans laquelle est écrit ce texte. Il s'agit d'un système qui procède en deux temps : d'abord nous effectuons une étape de reconnaissance de l'écriture, et à partir des sorties de cette étape nous effectuons une étape de classification de la langue. Pour réaliser ce système nous nous appuyons d'une part pour la reconnaissance sur notre système BLSTM-CTC (présenté dans le chapitre4), d'autre part pour l'identification de la langue sur un classifieur Bayesien naïf issu du module « Compact Language Detector 2 (CLD2) » [Sites].

Nous avons considéré deux possibilités pour ce système, soit la reconnaissance du texte est effectuée par plusieurs classifieurs dynamiques de type BLSTM-CTC, chacun spécialisé dans une langue, soit elle est effectuée par un seul classifieur dynamique capable de traiter toutes les langues. La figure 6.1 présente le système avec de multiples classifieurs dynamiques, tandis que la figure 6.2 présente le système avec un seul classifieur dynamique.

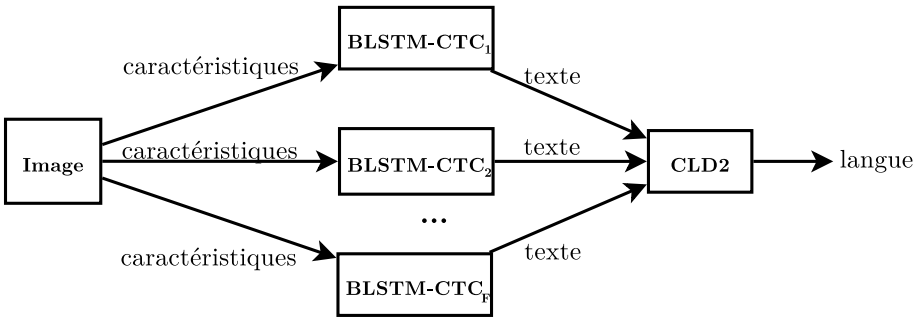


FIGURE 6.1 – Système avec F BLSTM-CTC chacun spécialisé sur une langue

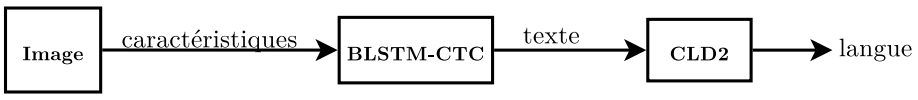


FIGURE 6.2 – Système avec un unique BLSTM-CTC spécialisé sur un script

Un classifieur dynamique spécialisé sur une langue est performant pour la reconnaissance de cette langue et permettra donc que l'étage de reconnaissance de la langue soit plus efficace pour reconnaître les N-grammes de caractères

les plus fréquents de cette langue, tandis qu'un classifieur dynamique optimisé pour l'ensemble des langues d'un script n'aura pas cette même performance, mais il disposera d'une meilleure généralisation sur l'ensemble des langues en moyenne.

Nous présentons maintenant notre système de reconnaissance de texte, puis nous présentons le fonctionnement de CLD2.

6.1.1.1 Système de reconnaissance de texte

Dans notre approche nous utilisons le classifieur dynamique « Bidirectional Long Short Term Memory – Connectionist Temporal Classification » (BLSTM-CTC) (voir section 2.2.3) pour effectuer la reconnaissance de texte, notre approche est similaire à celle décrite dans la section 4.1.1. Notre système de reconnaissance de l'écriture comporte une étape de prétraitement incluant une correction de l'inclinaison, une correction de la pente ainsi qu'une binarisation par seuil des images. Nous appliquons ensuite une méthode de fenêtres glissantes afin d'extraire les histogrammes de gradients décrivant une fenêtre (voir section 4.1.3.3). Nous avons utilisé les mêmes paramètres pour cette caractéristique que dans le chapitre 4 étant donné les résultats obtenus.

La séquence de vecteurs de caractéristiques extraite à partir d'une image est ensuite traitée par le BLSTM-CTC pour obtenir une séquence de vecteurs des probabilité a posteriori des caractères. Ces probabilités sont ensuite traitées afin de les transformer en séquence de caractères :

1. Pour chaque vecteur de probabilité nous sélectionnons le caractère avec la plus grande probabilité, nous obtenons ainsi une séquence des caractères les plus probables à chaque instant, par exemple

$$p = \{A, b, b, joker, joker, c, c, joker, c\} \tag{6.1}$$

   ;

2. De ce vecteur p nous supprimons les caractères identiques adjacents, $p = \{A, b, joker, c, joker, c\}$;

3. Nous supprimons ensuite tous les jokers, nous obtenons ainsi une séquence de caractères γ, par exemple $\gamma = Abcc$.

Cette séquence de caractères est ensuite traitée par CLD2 pour déterminer la langue contenue dans l'image.

6.1.1.2 Language Detector 2

CLD2 utilise un classifieur Bayesien naïf afin d'associer une langue à un texte. La classification est effectuée en mettant à jour la probabilité a posteriori de catégories avec des probabilités caractéristiques dans chaque catégorie :

$$p(C_k|X)^{m+1} \propto p(C_k|X)^m \,.\, p(X_i) << \tag{6.2}$$

avec C_k une catégorie (langue) et X le texte et X_i une caractéristique de ce texte et m l'itération. Les caractéristiques utilisées dépendent du script unicode détecté, dans le cas de script ne possédant qu'un langage il n'y a pas d'ambiguïté de classification, pour les scripts Han et Hangul les uni-grammes de symboles sont utilisés, pour les autres langues les quadri-grammes de caractères sont utilisés. Chaque langue possède un modèle associé, représentant les probabilités d'enchaînement des quadri-grammes. La ponctuation, les chiffres ainsi que les symboles ne sont pas utilisés pour classifier les langues. Selon ses développeurs, CLD2 offre les meilleures performances pour des chaînes de 200 caractères et plus.

Nous présentons maintenant les expériences réalisées afin de mesurer les performances de notre système de reconnaissance de la langue dans des documents manuscrits.

6.1.2 Expériences

Afin de mesurer la performance de notre système nous avons sélectionné deux scripts et deux langues par script. D'un côté nous avons le script bengali avec le bengali et l'assamais et de l'autre nous avons le script latin avec le français et l'anglais. Les langues du script latin diffèrent principalement par la fréquence des N-grammes de caractères (voir tableau 6.2), ainsi que par l'utilisation des caractères accentués en français. Concernant les caractères nous avons fait le choix de reconnaître uniquement les caractères (majuscules et minuscules), les caractères accentués et les espaces. Les symboles, les chiffres et la ponctuation n'apportent pas d'information pour la reconnaissance de la langue, ils sont donc ignorés. Cela signifie que pour le BLSTM-CTC ils appartiennent à la classe joker au moment de l'apprentissage, ce qui n'a pas d'influence sur la qualité de l'apprentissage. Au total nous avons 90 caractères pour le latin.

Dans le cadre de cette expérience, U. Garain, du laboratoire de vision et de reconnaissance de formes de l'institut de statistiques de Calcutta, nous a

gracieusement fourni des bases (qui seront rendues publiques pour ICDAR 2015) du script Bengali et nous a assisté dans la compréhension des règles qui régissent ce script. Le script bengali [1] est dit alphasyllabaire, c'est à dire qu'une voyelle est soit représentée par un caractère soit par un diacritique qui modifie le graphème de la consonante. De plus plusieurs consonantes peuvent être groupées pour créer un nouveau graphème. Le bengali et l'assamais diffèrent par l'utilisation de deux caractères supplémentaires en assamais. Au total nous avons dénombré plus de 4000 graphèmes différents. Nous avons néanmoins réduit ce nombre de graphèmes en supprimant un certain nombre de combinaisons impossibles ou inexistantes. Nous avons aussi séparé les voyelles diacritiques avec une composante verticale ne chevauchant pas la consonante marquée. Par exemple le graphème « akar », représenté sur la figure 6.3 est divisé en « ko » (la partie noire) et en diacritique du « a » (en rouge).

FIGURE 6.3 – Partitionnement du graphème « akar »

Cette étape de réduction des caractères nous a permis de réduire le nombre de sorties à 900 caractères, ce qui reste un nombre important de sorties, et pour lequel nous ne connaissons pas de précédent dans la littérature. Afin de mesurer les performances nous avons utilisé deux bases, l'une pour le latin, l'autre pour le bengali. Nous décrivons maintenant ces bases.

6.1.2.1 Bases

La base bengali comporte 2300 images avec 2100 images de bengali et 200 images d'assamais, ce qui représente respectivement 18000 mots et 2000 mots pour ces langues, Environ 1000 de ces images proviennent de la base présentée par R. Sarkar et al. [155], l'autre partie a été produite par l'Institut des Statistiques de Calcutta (ISC). Cette base a été divisée en plusieurs sous-bases avec les ratios suivants : 60% pour l'apprentissage, 20% pour la validation et 20% pour le test. Étant donné la faible quantité de données pour l'assamais nous n'avons pas réalisé une spécialisation du système de reconnaissance sur ces langues. Il est important de noter que CLD2 ne possède pas de modèles de

1. Attention le mot « bengali » désigne à la fois le script et une langue de ce script, par la suite nous précisons « script bengali » lorsque nous parlons du script.

langue pour l'assamais, nous avons donc appris un modèle à partir d'un corpus assamais fourni par l'ISC.

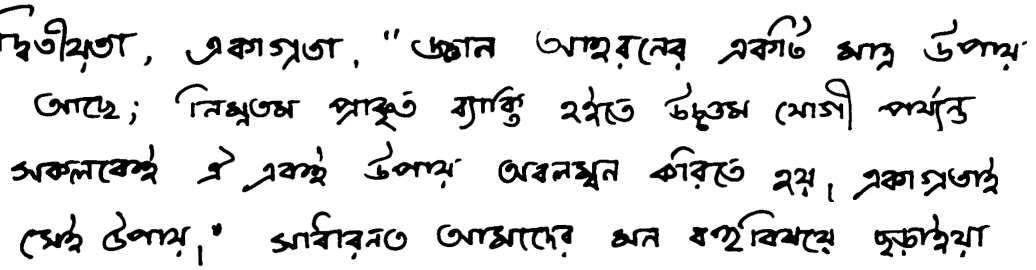


FIGURE 6.4 – Images de lignes Bengali

Pour le script latin nous avons utilisé la base du Projet d'Études Amont (PEA) Moyens AUtomatisés pour la Reconnaissance de DOcuments écRits [DGA et al.](MAURDOR). La base MAURDOR comporte près de 10000 documents factices. Les documents sont rédigés en trois langues majoritaires : français, anglais et arabe. D'autres langues peuvent apparaître de façon très minoritaire dans le corpus (mois de 1%) mais elles ne sont pas annotées. Les documents contiennent des zones de textes manuscrits et dactylographiés, des graphiques, des tableaux, des photographies, etc.. Ces documents ont été créés par plus de 1000 personnes différentes, chaque personne recevant au plus dix scénarios de création de documents parmi plus de 1000 scénarios existants. Ces scénarios peuvent être partitionnés en plusieurs types :

C1 : formulaires imprimés et remplis à la main ;

C2 : documents commerciaux, privés ou personnels (devis, reçu, chèque, etc.) ;

C3 : correspondances privées manuscrites sur papier libre ou à entête (carte postale, post-it, etc.) ;

C4 : correspondances privées ou personnelles dactylographiées (fax, mél, courrier, note de service, etc.) ;

C5 : documents autres (plan dessiné à main levé, extrait de code, échec de numérisation, etc.).

La figure 6.5 présente différents documents présents dans le corpus MAURDOR. Le corpus ainsi constitué possède une hétérogénéité importante, avec une grande variabilité sur le fond et la forme.

Nous avons utilisé les zones de textes identifiées dans la base d'apprentissage de MAURDOR afin de constituer nos bases. Nous avons au total 20000

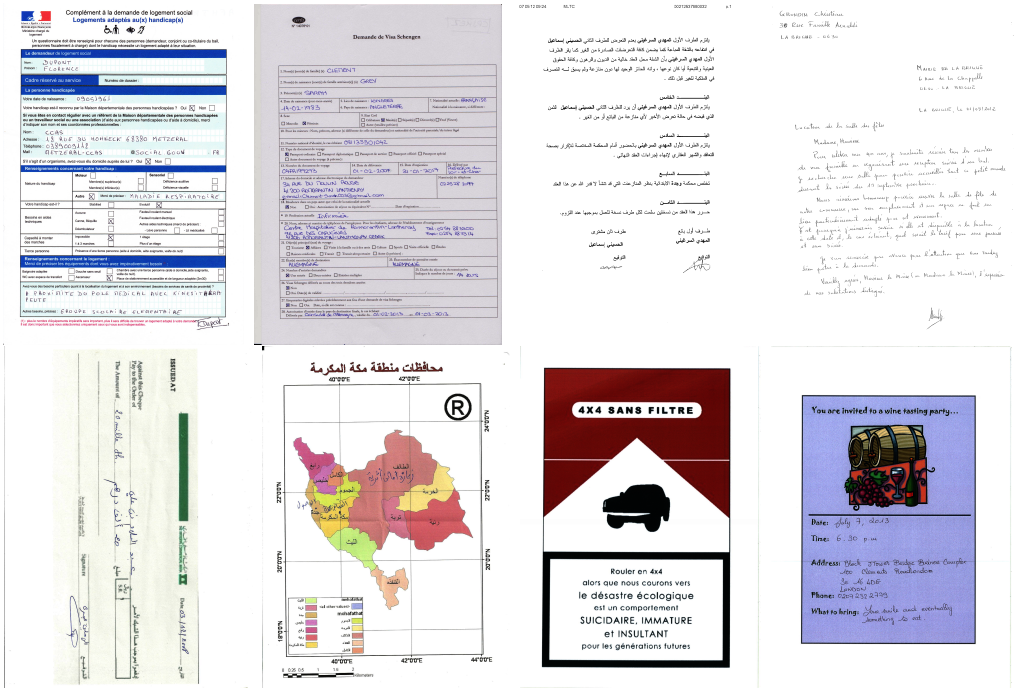


FIGURE 6.5 – Documents du corpus MAURDOR

lignes de français et 8700 lignes d'anglais, avec environ 53000 et 20000 mots respectivement pour ces langues.

Afin de mesurer l'impact de la spécialisation de notre classifieur dynamique effectuant la reconnaissance de texte nous avons généré trois bases : une base ne contenant que le français, une base ne contenant que l'anglais, une base contenant à la fois le français et l'anglais. Ces bases ont été divisées en plusieurs sous-bases avec les ratios suivants : 60% pour l'apprentissage, 20% pour la validation et 20% pour le test.

Nous présentons maintenant les résultats obtenus sur ces différentes bases.

6.1.2.2 Résultats

Nous présentons dans un premier temps les résultats obtenus en terme de reconnaissance de l'écriture sur ces différentes bases dans le tableau 6.1. Les systèmes sont appris sur une langue puis validés et testés sur l'ensemble des langues du corpus. Pour ces résultats nous présentons l'erreur brute en sortie du BLSTM-CTC sans modèle de langue, nous présentons deux mesures de performances pour nos systèmes :

— l'erreur au niveau caractères (« Character Error Rate » (CER)) qui est la distance d'édition relative (avec des pondérations égales entre la substitution, l'insertion et la délétion) de la chaîne de caractères obtenue

en sortie du BLSTM-CTC avec la chaîne de caractères idéale ;

— l'erreur au niveau de la séquence (« Sequence Error Rate » (SER)) qui représente le pourcentage de chaînes de caractères mal reconnues, avec au moins une erreur de caractère générée par le BLSTM-CTC.

Système	Base de test	CER	SER
Anglais	Latin	46.37%	90.45%
Français	Latin	35.22%	84.00%
Latin	Latin	30.69%	80.69%
Bengali	Script bengali	24.60%	99.40%

TABLE 6.1 – Résultats des différents systèmes de reconnaissance de texte

Les performances des BLSTM-CTC sont mesurées sur une base comportant les deux langues traitées.

Nous constatons dans un premier temps une différence importante entre les résultats des systèmes appris sur du français et de l'anglais, cette différence s'explique par la quantité de données. Le système appris sur le français a deux fois plus de données et est donc meilleur en généralisation.

Dans un second temps nous pouvons voir que le système appris sur le latin ne souffre pas de la différence linguistique entre le français et l'anglais, et notamment de la différence entre les fréquences des uni-grammes ou bi-grammes de caractères des deux langues. Pour rappel le tableau 6.2 présente la fréquence des 10 uni-grammes et bi-grammes les plus courants dans les deux langues. Le BLSTM-CTC latin est donc capable de généraliser de manière efficace sur les deux langues.

Nous présentons maintenant les résultats obtenus pour la reconnaissance de la langue, à l'aide des figures suivantes :

1. Figure 6.6 : résultats de la reconnaissance de langue avec le BLSTM-CTC appris sur la base contenant uniquement du français ;

Français				Anglais			
Uni-gramme	Fréquence	Bi-gramme	Fréquence	Uni-gramme	Fréquence	Bi-gramme	Fréquence
E	17.35%	ES	3.05%	E	12.49%	TH	3.56%
A	8.2%	LE	2.22%	T	9.28%	HE	3.07%
S	7.93%	DE	2.17%	A	8.04%	IN	2.43%
I	7.53%	RE	2.1%	O	7.64%	ER	2.05%
N	7.17%	EN	2.08%	I	7.57%	AN	1.99%
T	6.99%	ON	1.64%	N	7.23%	RE	1.85%
R	6.65%	NT	1.62%	S	6.51%	ON	1.76%
L	5.92%	ER	1.53%	R	6.28%	AT	1.49%
U	5.73%	TE	1.52%	H	5.05%	EN	1.45%
O	5.53%	ET	1.43%	L	4.07%	ND	1.35%

TABLE 6.2 – Fréquence des 10 Uni-grammes et bi-grammes de caractères les plus fréquents en anglais et en français

2. Figure 6.7 : résultats de la reconnaissance de langue avec le BLSTM-CTC appris sur la base contenant uniquement de l'anglais ;
3. Figure 6.8 : résultats de la reconnaissance de langue avec le système à deux BLSTM-CTC, avec une décision de l'expert (affectation à l'anglais réalisée à partir des sorties provenant du BLSTM-CTC spécialisé sur l'anglais et affectation au français réalisée à partir des sorties provenant du BLSTM-CTC spécialisé sur le français) ;
4. Figure 6.9 : résultats de la reconnaissance de langue avec le BLSTM-CTC appris sur la base latin ;
5. Figure 6.10 : résultats de la reconnaissance de langue avec le BLSTM-CTC appris sur la base du script bengali.

Pour chaque BLSTM-CTC nous présentons trois courbes qui représentent le score d'identification de la langue en fonction de la longueur de la chaîne de caractères. La courbe bleue représente les résultats de CLD2 sur la vérité terrain, tandis que la courbe verte représente les résultats de CLD2 pour la chaîne de caractères produits par le BLSTM-CTC, nous pouvons ainsi nous comparer aux résultats produits par CLD2 sur la vérité terrain et mesurer la perte occasionnée pour des documents manuscrits. Les courbes présentées sont :

1. les résultats sur les images de texte de la base de test appartenant à l'une des langues du script ;
2. les résultats sur les images de texte appartenant à l'autre langue considérée pour ce script ;
3. les résultats sur les deux langues du script de la base de test.

D'une manière générale nous pouvons confirmer que CLD2 offre de meilleures performances pour une séquence plus longue, ce qui confirme les indications des auteurs de CLD2. De plus remarquons clairement que lorsqu'un BLSTM-CTC n'est pas appris sur une langue il offre des performances très médiocre pour l'identification de cette langue, comme c'est le cas pour le BLSTM-CTC appris sur du français et devant reconnaître de l'anglais et vice-versa. Nous faisons l'hypothèse que les BLSTM-CTC apprennent des enchaînements de caractères (bi-grammes ou supérieurs) propre à cette langue, puisque la fréquence des uni-grammes entre les deux langues est assez similaire. Le BLSTM-CTC appris sur le script latin offre des performances de reconnaissance de la langue légèrement plus faible que les BLSTM-CTC spécialisés sur une langue, cependant la distinction entre les deux langues est meilleure. Si pour ces différents

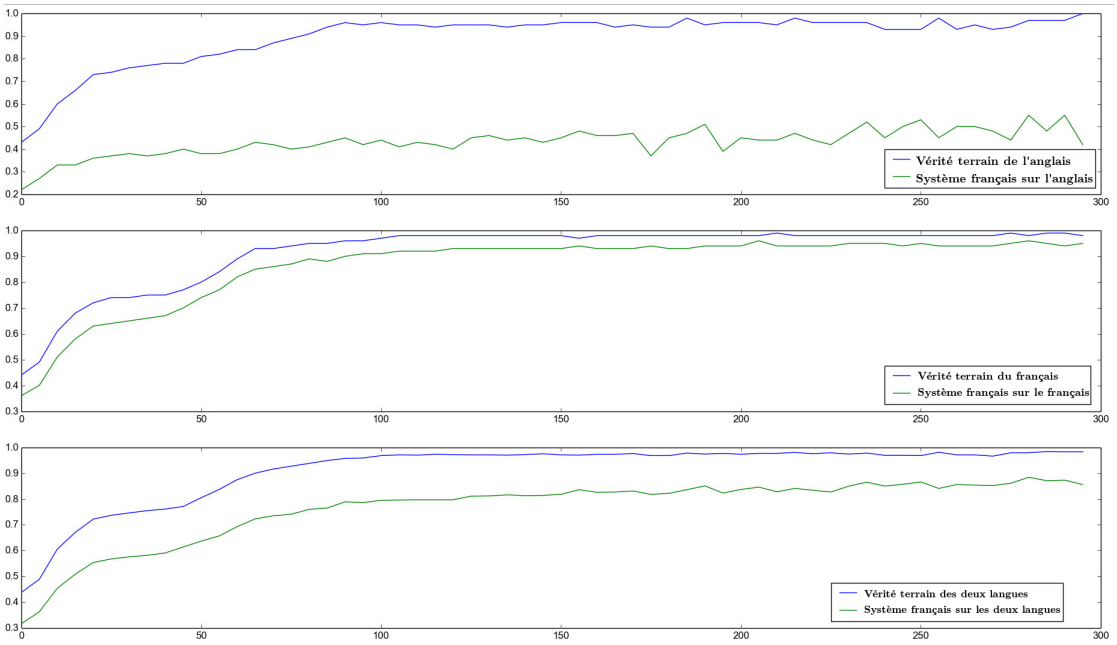


FIGURE 6.6 – Résultats de reconnaissance de langue avec le BLSTM-CTC appris sur la base contenant uniquement du français

systèmes nous analysons les écarts entre la vérité terrain sur la base latin et les résultats du système nous avons entre 15 à 20% pour le système avec BLSTM-CTC français, de 25 à 45% pour le système avec BLSTM-CTC anglais, de 15 à 25% pour le système avec décision de l'expert, et de 10 à 15% pour le système avec BLSTM-CTC latin. Pour notre problème d'identification de la langue, il est donc plus pertinent d'apprendre un unique BLSTM-CTC sur l'ensemble des langues du script, et d'utiliser ses sorties pour effectuer l'identification de la langue, que de spécialiser de multiples BLSTM-CTC sur plusieurs langues, c'est le résultat observé sur les courbes 6.9 et 6.8. Bien que le BLSTM-CTC intègre à l'aide de sa mémoire des composantes linguistiques nous pouvons voir que celà ne nuit pas de manière importante à la qualité de la reconnaissance d'une langue ou d'une autre.

Le tableau 6.3 présente un certain nombre d'images ainsi que le résultat texte obtenu et la langue identifiée. En analysant les erreurs nous pouvons voir que certaines de nos erreurs sont liées à des images relativement complexes, pour lesquelles les prétraitements ou les caractéristiques ne sont pas adaptés et qu'il s'agit de textes très courts.

Les résultats de reconnaissance des langues du script bengali sont difficiles à juger. D'un côté la reconnaissance de la langue bengali est très bonne avec un plateau à 99% pour plus de 260 caractères, de l'autre la reconnaissance de l'assamais est très mauvaise, à peine 25% pour un nombre de caractères

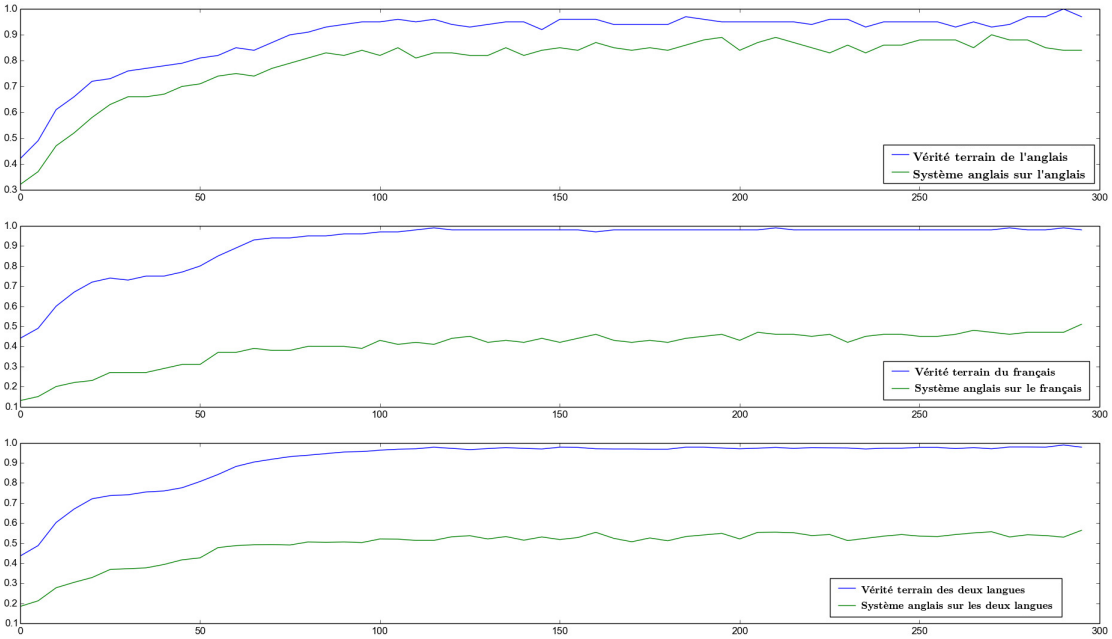


FIGURE 6.7 – Résultats de la reconnaissance de langue avec le BLSTM-CTC appris sur la base contenant uniquement de l'anglais

égal à 12. Les mauvais résultats de cette expérience peuvent être expliqués par l'absence d'exemples d'assamais pour l'apprentissage du BLSTM-CTC ainsi qu'une différence trop importante entre les données et le corpus utilisé pour l'apprentissage du modèle d'assamais pour CLD2.

Pour les deux scripts nous pouvons dire que la majorité des erreurs sont liées aux erreurs de reconnaissance du BLSTM-CTC, d'autres sont liées à CLD2 qui supprime de la chaîne de caractères les groupes de caractères entièrement en majuscules (par exemple des références ou des sigles) et les groupes de caractères représentant des noms propres. Si une chaîne de caractères comporte plusieurs de ces groupes de caractères et très peu de mots propres à une langue l'identification de la langue devient plus difficile.

6.1.2.3 Conclusion

Nous avons présenté une approche pour la reconnaissance de la langue dans une image de texte manuscrit. Notre approche se distingue des autres car elle ne s'appuie pas sur une classification de caractéristiques extraites d'une image, mais sur une reconnaissance de texte sans appui linguistique et une classification par l'étude des quadri-grammes de caractères. Dans le cas de l'écriture manuscrite, la différence majeure entre différentes langues d'un même script ne sera pas nécessairement visible au travers des caractéristiques provenant

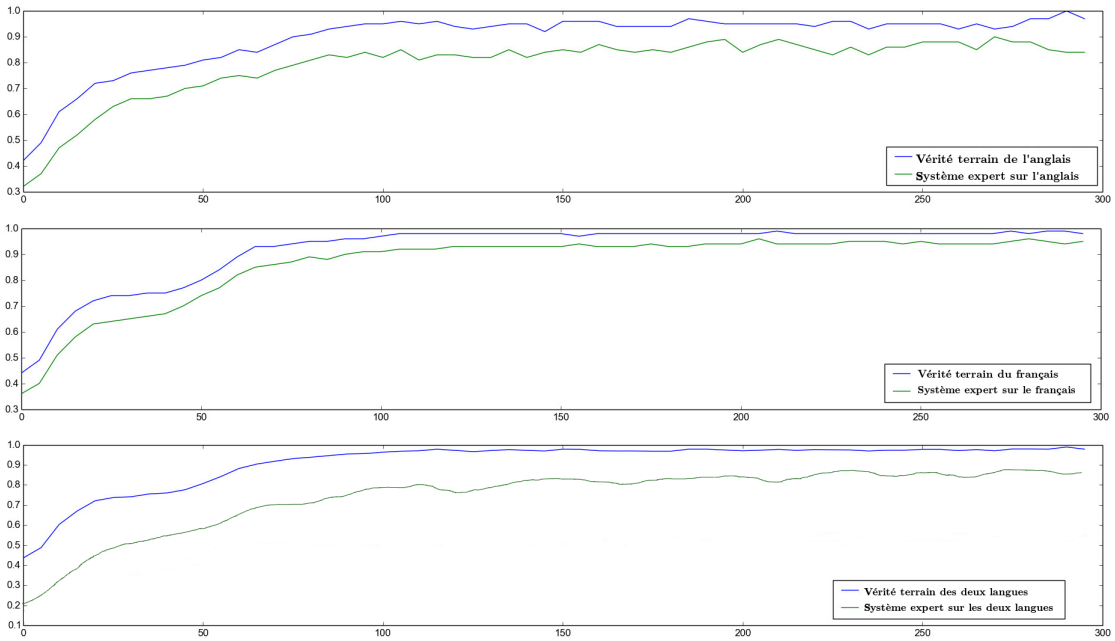


FIGURE 6.8 – Résultats de la reconnaissance de langue avec les BLSTM-CTC mono-langues et une décision de l'expert.

directement de l'image car la morphologie des caractères ne change pas de manière importante entre les scripts, mais elle varie de manière importante entre les scripteurs. Avec une méthode basée sur la classification de caractéristiques on risque d'apprendre des caractéristiques des scripteurs de la base et non des caractéristiques de la langue. En revanche les fréquences d'apparitions des quadri-grammes changent d'une langue à une autre. L'utilisation des quadri-grammes nous permet d'utiliser des informations plus pertinentes pour l'identification des langues. Le système que nous avons mis en place permet d'extraire un texte depuis une image, pour ensuite utiliser cette sortie pour identifier la langue. Les résultats de ce système sont donc très dépendant de la qualité de la reconnaissance de texte.

Si nous comparons notre système à celui de J.Hochberg [93], bien que nos résultats bruts soient inférieurs en réalité notre système est sans doute plus performant. En effet dans [93] de nombreux pré-traitements manuels sont réalisés, les documents sont très propres et les documents d'apprentissage sont identiques à ceux de test. Le système présenté est donc sur-optimisé sur une base simple. Notre système ne possède pas ces défauts, il est donc plus probable qu'il soit meilleur sur des données réelles.

Nous avons pu montrer au travers de nos expériences que le BLSTM-CTC apprend un modèle de langage sur lequel il s'appuie durant la reconnaissance. Ce modèle de langage permet d'être plus performant pour identifier une langue

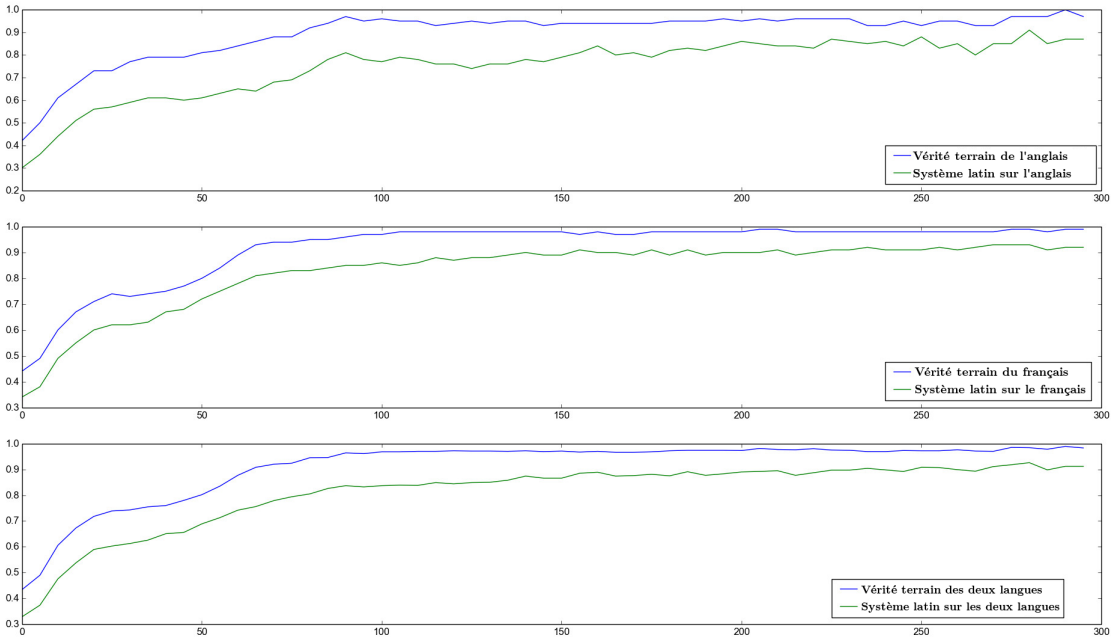


FIGURE 6.9 – Résultats de la reconnaissance de langue avec le BLSTM-CTC appris sur la base latin

parmi un ensemble d'autres langues. En revanche dans le cas ou nous désirons identifier plusieurs langues d'un même script les meilleures performances sont observées en utilisant un BLSTM-CTC appris sur l'ensemble des langues de ce script. Nous avons pu observer ce résultat pour deux langues, il est donc important de généraliser ce résultat à l'aide de nouvelles bases multi-lingues (en dehors de MAURDOR il n'existe pas à notre connaissance de bases multi-lingues produites dans des conditions hétérogènes). En effet l'apprentissage d'un unique BLSTM-CTC pour plusieurs langues produit un système de reconnaissance très performant au niveau caractère, cependant nous ne pouvons pas inférer la qualité du modèle de langue appris de manière implicite par le BLSTM-CTC pour un grand nombre de langues. L'apprentissage d'un plus grand nombre de langues pourrait n'avoir aucun effet négatif pour la reconnaissance global comme il pourrait le dégrader. Dans ce cas il semble plus intéressant d'utiliser une approche « one against all » pour l'identification de la langue, c'est à dire spécialiser les BLSTM-CTC sur une langue et comparer les probabilités a posteriori des langues produites par les systèmes. La sortie texte ayant la meilleure probabilité a posteriori sur une langue sera affecté à cette langue.

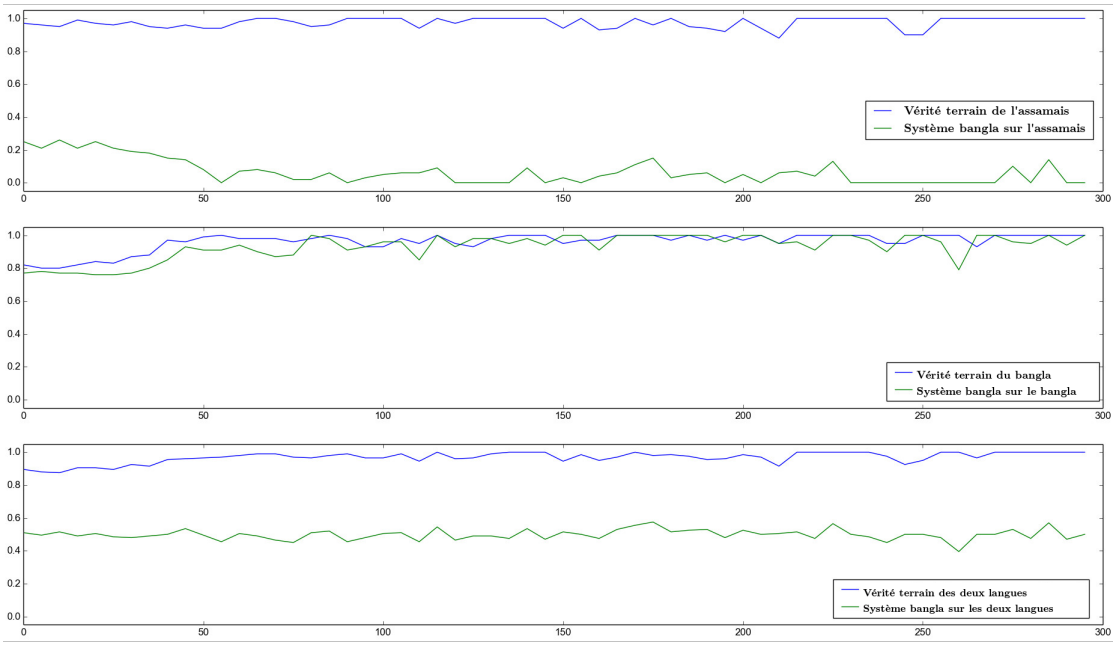


FIGURE 6.10 – Résultats de la reconnaissance de langue avec le BLSTM-CTC appris sur la base du script bengali

6.2 Détection de mots clefs

La recherche d'un mot clef, d'une expression régulière ou d'une requête dans un document ou un corpus de documents, est extrêmement importante car elle permet à un utilisateur d'accéder plus rapidement au contenu informationnel qu'il désire. Sur un texte digital il est très facile de trouver à l'aide d'un mot clef ou d'une expression régulière l'ensemble des zones comportant ce mot ou cette expression régulière. Accomplir une recherche similaire dans un fichier audio [68, 79] ou un fichier image[55] n'est pas possible sans avoir une transcription exacte du contenu de ce fichier. Étant donné la masse grandissante de documents provenant de différents média il est intéressant de pouvoir accéder à l'ensemble des informations contenus dans ces documents, il est donc nécessaire de produire une transcription. Dans cette section nous nous intéressons aux méthodes permettant d'effectuer une recherche de mots clefs dans une image de document numérisé [55].

Il existe deux grandes catégories de méthodes pour effectuer la détection de mots clefs. Soit il s'agit d'une recherche à partir d'une requête textuelle, soit il s'agit d'une requête à partir d'une image.

La recherche à partir d'une requête textuelle [66, 67, 101, 175, 177] est possible dans le cas où le système de reconnaissance de texte est performant et permet d'extraire de manière quasi-parfaite l'ensemble du texte contenu dans

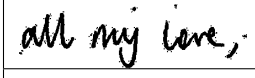
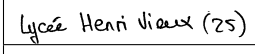
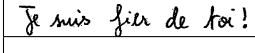
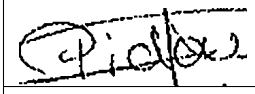
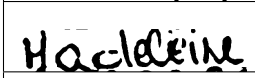
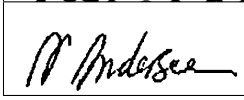
Image originale	Texte reconnu	Langue obtenue	Langue réelle
	all my love	anglais	anglais
	Lycée Henri Viauv	français	français
	Je suir fier de toi	français	français
	Diallou	italien	anglais
	Haclexirie	roumain	français
	R Ndesee	afrikaans	anglais

TABLE 6.3 – Résultats de reconnaissance de la langue

les images. Il est donc nécessaire d'avoir un classifieur dynamique à l'état de l'art ainsi qu'une base d'apprentissage importante pour permettre au classifieur de généraliser et d'être performant.

Dans le cas où nous disposons d'un corpus de documents numérisés et dont le texte a été extrait par un système de Reconnaissance Optique de Caractères (ROC), il semble assez simple de trouver un mot clef à l'aide d'un algorithme de recherche de sous-chaînes de caractères [30, 98]. Cependant le texte extrait par des systèmes ROC n'est jamais sans fautes, on peut alors penser qu'une recherche de mots clef à l'aide d'un algorithme de recherche de sous-chaînes inexactes de caractères [76, 88] peut permettre de détecter le mot clef désiré. Ces approches ne sont pas suffisantes sur des textes complexes, notamment manuscrits, pour lesquels les erreurs de reconnaissance sont trop importantes. Une approche proposée par S. Thomas, C. Chatelain, L. Heutte et T. Paquet[175] s'appuie sur l'utilisation d'un Modèle de Markov Caché (MMC) pour la détection de mots clefs pendant la reconnaissance (le mot clef est recherché sur un texte dont le contenu n'a pas été extrait). Le MMC possède une topologie particulière présentée sur la figure 6.11. Il constitue un modèle d'extraction. Ce modèle comporte des modèles de remplissage, qui servent à absorber l'ensemble des caractères n'appartenant pas au mot clef, et le mot clef, afin de forcer le MMC à détecter ce mot clef dans la chaîne. Le modèle de remplissage est un modèle ergodique de transitions sur l'ensemble des caractères existants, voir figure 6.12. Le fait de forcer le modèle à passer dans le mot clef recherché permet d'obtenir, à l'aide de l'inférence de Viterbi, la vraisemblance de ce mot dans un texte. Afin de décider si le mot est présent un ratio entre la

vraisemblance avec la topologie présentée en figure 6.11 et une topologie avec uniquement un modèle de remplissage est calculé. Si ce ratio est supérieur à un seuil alors le mot clef recherché est présent dans le texte.

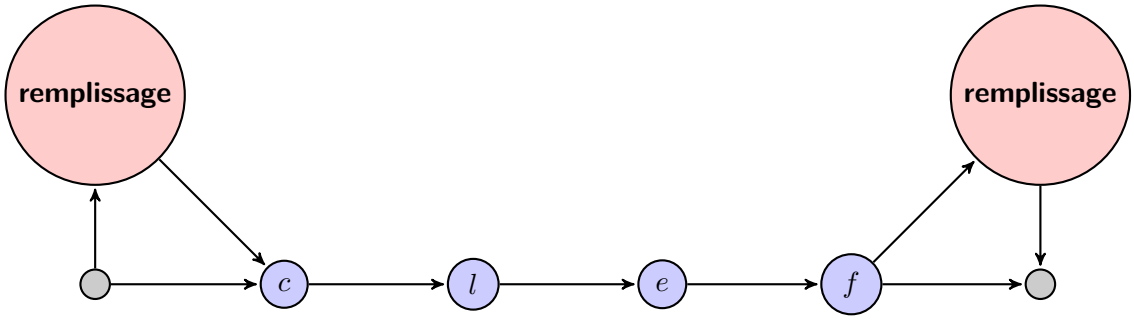


FIGURE 6.11 – Topologie du MMC utilisé pour la détection de mots clefs

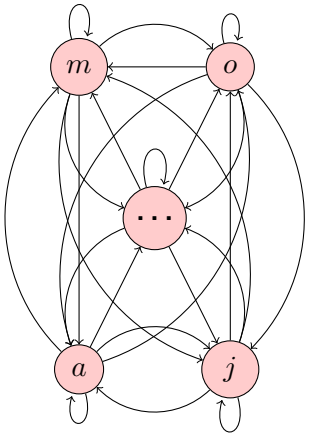


FIGURE 6.12 – Modèle ergodique de remplissage

La recherche à partir d'une image [22, 106, 115, 146] permet d'effectuer des recherches dans le cas où aucun système de reconnaissance de texte n'est disponible ou assez performant, par exemple sur des textes anciens, ou des documents fortement dégradés. Cette recherche s'effectue de manière générale en extrayant des caractéristiques dans l'image de la requête et en effectuant une recherche de zones dans les images des documents ayant des caractéristiques similaires. La recherche par similarité d'images permet donc de combler les déficits des systèmes de reconnaissance sur des documents complexes, cependant elle est plus coûteuse en temps d'exécution car chaque image requête doit être comparée à l'ensemble du corpus de recherche, alors que la requête textuelle est bien plus rapide (à condition que la reconnaissance de texte du corpus soit réalisée au préalable).

Dans cette section nous présentons nos travaux réalisés sur les données de la compétition ICDAR 2015 de détection de mots clefs. Cette compétition est intéressante car elle permet de comparer les deux approches de détection de mots. Dans un premier temps nous présentons notre approche pour la détection de mots clefs, dans un second temps nous présentons les expériences et les résultats obtenus sur la base d'évaluation intermédiaire. Les résultats finaux n'étant pas connus à ce jour.

6.2.1 Systèmes de détection de mots clefs

Nous présentons trois systèmes de détection de mots clefs, deux de ces systèmes traitent des requêtes images, le troisième traite des requêtes textuelles. Nos systèmes s'appliquent en deux temps, dans un premier temps nous extrayons à l'aide d'un classifieur dynamique, le BLSTM-CTC dans notre cas, les probabilités a posteriori des caractères sur notre corpus de texte (indexation des images par des treillis de caractères), dans un second temps nous traitons les requêtes. La figure 6.13 représente l'indexation des images.

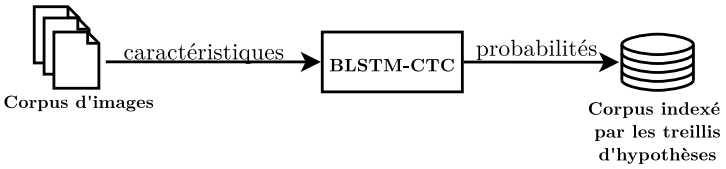


FIGURE 6.13 – Indexation du corpus de documents

Dans un premier temps nous présentons l'indexation des documents [154], c'est à dire le processus qui permet d'obtenir les probabilités a posteriori des caractères, puis nous présentons nos trois systèmes de détection de mots clefs.

6.2.1.1 Indexation des documents

Étant donné les performances du BLSTM-CTC obtenues dans le chapitre 4 pour la reconnaissance de l'écriture manuscrite nous l'utilisons de nouveau dans ce système pour réaliser l'indexation de notre corpus de documents. Notre système de reconnaissance de l'écriture comporte une étape de prétraitement incluant une correction de l'inclinaison, une correction de la pente ainsi qu'une binarisation adaptative gaussienne des images [179]. Nous appliquons ensuite une méthode de fenêtres glissantes afin d'extraire différentes caractéristiques : Caractéristiques des Pixels (CDP) (voir section 4.1.3.2), les HoG (voir section 4.1.3.3) et « Oriented FAST and Rotated BRIEF » (ORB) (voir section 4.1.3.4)

. Nous avons utilisé les mêmes paramètres pour ces caractéristiques que dans les chapitres précédents étant donné les résultats obtenus sur une base précédente. Nous n'avons pas inclus les Pixels, car malheureusement il n'ont pas permis d'obtenir des performances suffisantes durant nos premiers essais, et ce malgré des adaptations de paramètres.

La séquence de vecteurs de caractéristiques extraite à partir d'une image est ensuite traitée par le BLSTM-CTC pour obtenir une séquence de vecteurs des probabilités a posteriori des caractères. Dans cette application nous comparons plusieurs systèmes BLSTM-CTC :

1. Système mono-caractéristique avec les CDP ;
2. Système mono-caractéristique avec les HoG ;
3. Système mono-caractéristique avec les ORB ;
4. Système de combinaison multi-caractéristiques de bas niveau (voir section 5.1.1) avec les CDP et HoG ;
5. Système de combinaison multi-caractéristiques de haut niveau (voir section 5.1.3) en appliquant un opérateur de combinaison moyen sur les sorties des BLSTM-CTC spécialisés sur les caractéristiques CDP, HoG et ORB , puis en apprenant un nouveau BLSTM-CTC sur les sorties moyennes ;

Nous avons choisi d'utiliser la combinaison bas niveau avec les caractéristiques HoG et CDP car elle offre des performances satisfaisantes ainsi qu'un temps d'apprentissage et d'exécution rapide. Nous avons aussi pris la combinaison de haut niveau avec une moyenne des caractéristiques et un BLSTM-CTC car avec l'utilisation d'un dictionnaire, c'est le système de combinaison le plus performant. Nous allons ainsi pouvoir mesurer l'apport de nos différentes combinaisons sur une application. Après avoir obtenu les probabilités a posteriori sur notre corpus d'images, nous pouvons désormais traiter les requêtes en décodant le treillis d'hypothèses fourni par le BLSTM-CTC à l'aide du modèle MMC d'extraction.

6.2.1.2 Requête textuelle

Les requêtes textuelles sont traitées par le MMC présenté dans [19, 175] qui les représente sous un enchaînement d'états spécifiques. Cet enchaînement d'état correspond à un modèle de remplissage, suivi du mot recherché, puis un autre modèle de remplissage, voir figure 6.11. Ce MMC parcourt l'ensemble du corpus texte dans le but de trouver une zone de texte avec une forte probabilité

d'être la requête. La figure 6.14 présente le système de détection de mots clefs pour une requête textuelle.

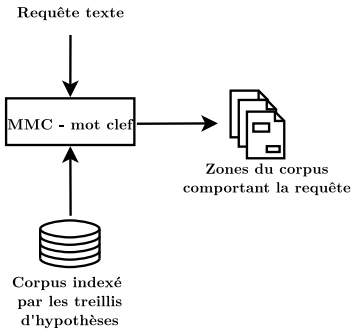


FIGURE 6.14 – Système de détection de mots clefs avec une requête textuelle

Étant donné le signal produit par le BLSTM-CTC, c'est à dire avec des probabilités a posteriori des caractères proche d'un Dirac, nous avons choisi de représenter les caractères par un seul état.

Le décodage d'une ligne de texte est effectué avec l'inférence Viterbi afin d'obtenir la séquence de caractères ayant le maximum de vraisemblance. Nous utilisons les probabilités a posteriori du BLSTM-CTC afin d'obtenir le score pour accepter ou rejeter une hypothèse. Le score d'une hypothèse est calculé comme la moyenne des probabilités a posteriori des caractères divisée par le nombre de trames sur lesquelles s'étend l'hypothèse. Ce score est ensuite normalisé en divisant par le nombre de caractères de la requête. Si le score est supérieur à un seuil, alors l'image contient le mot recherché, sinon elle ne le contient pas.

Nous nous appuyons sur les décisions du BLSTM-CTC pour produire des probabilités a posteriori, le MMC modélise une séquence et permet d'utiliser des informations linguistiques (modèle de langage ou dictionnaire) pour éventuellement corriger des erreurs lors de la reconnaissance.

Nous présentons maintenant seconde une approche pour traiter des requêtes images avec le même système.

6.2.1.3 Requête image

Notre premier système de détection de mots clefs à partir de requêtes images, transforme dans un premier temps cette requête image en une requête textuelle à l'aide d'un classifieur dynamique BLSTM-CTC, puis nous utilisons ce résultat comme requête textuelle. Il s'agit d'un système original dont le fonctionnement n'a à notre connaissance pas été abordé dans littéra-

ture. La figure 6.15 présente ce système. La transformation des probabilités a posteriori en texte se fait de manière identique à la méthode présentée dans la section 6.1.1.1. Cette approche n'est pas usuelle par rapport à la littérature du domaine [22, 106, 115, 146], mais étant donné le problème de détection de mots clefs il parait plus logique d'utiliser une requête textuelle et nous décidons donc d'explorer cette piste. Une piste similaire est explorée par G. Chen, C. Parada et T. N. Sainath [41] dans le domaine de l'audio.

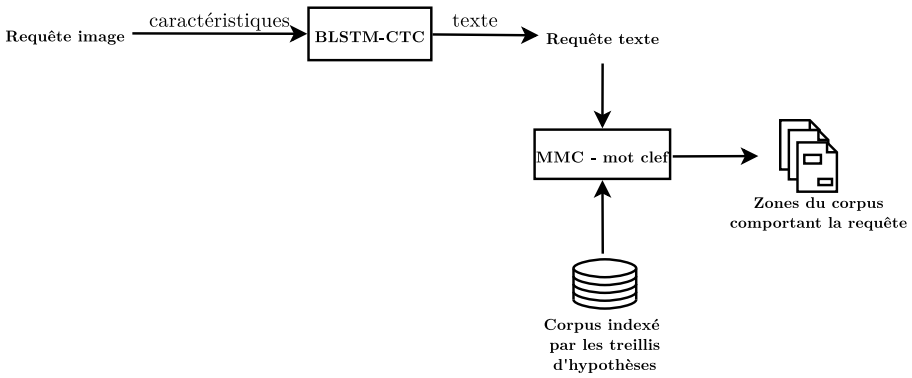


FIGURE 6.15 – Système de détection de mots clefs avec une requête image

6.2.1.4 Requêtes image avec correction

Le système précédent produira nécessairement des erreurs lors de la transformation de la requête image en requête texte, nous proposons donc dans notre second système de détection de mots clefs à partir de requêtes images, d'effectuer la même transformation et d'ajouter une étape de correction linguistique à l'aide d'un MMC effectuant un décodage dirigé par le lexique. La figure 6.16 présente ce système. Pour effectuer cette correction le MMC comporte un état par caractère et utilise un algorithme de Viterbi. La requête textuelle que nous utilisons appartient donc au lexique du corpus, ce qui devrait améliorer les résultats lors de la recherche, cependant elle ne permet pas de rechercher des mots hors-vocabulaire, contrairement au système précédent.

Nous présentons maintenant les expériences réalisées sur nos systèmes.

6.2.2 Expériences

Dans ces expériences nous comparons tout d'abord nos différents BLSTM-CTC sur le système de requêtes textuelles, puis en utilisant le BLSTM-CTC le plus performant sur cette application nous comparons les différentes approches

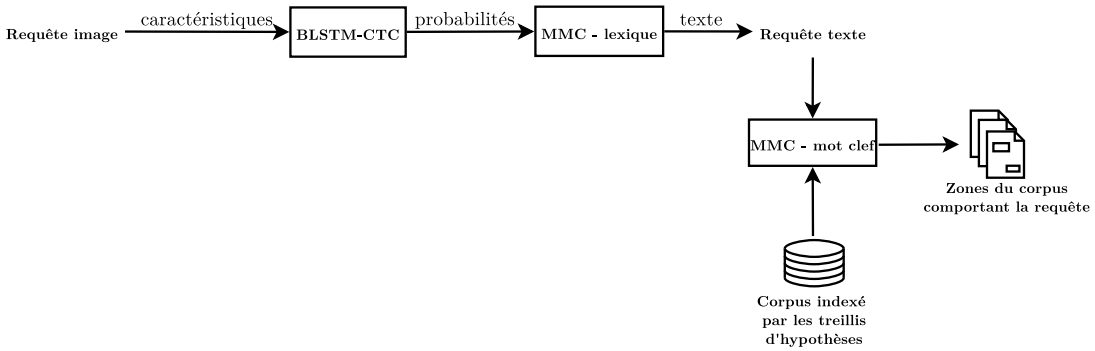


FIGURE 6.16 – Système de détection de mots clefs avec une requête image, et une correction du texte de la requête

de détection de mots clefs, c'est à dire à partir d'une requête image transformée en requête textuelle, à partir d'une requête image transformée en requête textuelle avec un décodage dirigé par le lexique et à partir d'une requête textuelle. Ces approches sont présentées respectivement sur les figures 6.15, 6.16 et 6.14.

6.2.2.1 Base

Nous utilisons la base mise à disposition lors de la compétition de détection de mot clefs pour ICDAR 2015. Il s'agit d'une série de lignes appartenant à la collection de documents Bentham, c'est à dire des lignes produites par Jeremy Bentham (1742-1832) ou son équipe de secrétaires. Il s'agit de documents manuscrits comportant de nombreuses ratures, annotations et marginalia. Ces documents sont donc plus complexes que ceux de la base RIMES que nous avons utilisé dans les chapitres précédents. Nous avons au total 11140 images de lignes, avec environ 104000 mots et 105 caractères différents. La figure 6.17 présente plusieurs exemples d'images de cette base. Nous avons divisé cette base en deux avec 80% de lignes pour la base d'apprentissage et 20% pour la base de validation. Nous n'avons pas de base de test car celle ci nous a été fournie par les organisateurs après que nous ayons produit les résultats présentés dans la thèse.

6.2.2.2 Résultats

Dans un premier temps nous présentons les résultats obtenus par nos différents BLSTM-CTC sur la reconnaissance de texte, ces résultats sont compilés dans le tableau 6.4.

De manière générale, les BLSTM-CTC offrent des performances très satis-

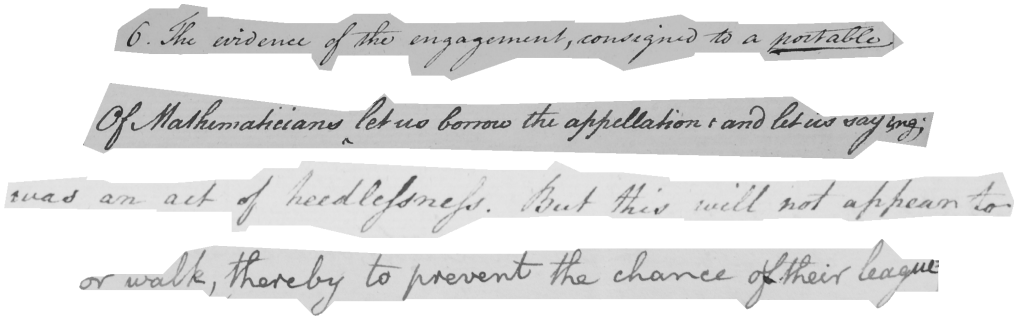


FIGURE 6.17 – Images de lignes de Bentham

Système	Caractéristiques	CER	SER
Mono-caractéristique	CDP	15.74%	93.81%
Mono-caractéristique	HoG	14.25%	93.17%
Mono-caractéristique	ORB	13.87%	92.45%
Multi-caractéristique, bas niveau	HoG et CDP	12.48%	90.33%
Multi-caractéristiques, haut niveau (moyenne et CTC)	ORB, HoG et CDP	9.39%	82.28%

TABLE 6.4 – Résultats de reconnaissance des BLSTM-CTC sur la base Bentham

faisantes pour la reconnaissance de l'écriture, le CER est très bas sur l'ensemble des systèmes. Ces performances élevées sont liées à la régularité de l'écriture de cette base et à nos prétraitements qui permettent de normaliser ces variations. Le système de combinaisons multi-caractéristiques haut niveau (moyenne et CTC) a des performances supérieures aux autres systèmes, il semble être un bon candidat pour la reconnaissance de texte dans cette application.

Nous vérifions cela en mesurant les performances de ces différents BLSTM-CTC en effectuant les requêtes textuelles de mots clefs. Nous avons effectués les vingt requêtes textuelles proposées par les organisateurs de la compétition, pour la validation de notre système. Parmi ces requêtes nous avons les mots suivants : « amount », « Bentham », « majesty », « parliament », « place », etc.. La décision d'avoir ces requêtes ou non dans une chaîne de caractères est effectuée avec un seuil de décision sur le maximum de vraisemblance obtenue via le MMC. Choisir le bon seuil est donc très important : avec un seuil trop faible nous obtenons un nombre de faux positifs trop grand et avec un seuil trop grand nous ne détecterons pas de vrais positifs. Pour un seuil donné nous obtenons sur l'ensemble des mots clefs un nombre de vrais positifs (VP), de faux positifs (FP) et de faux négatifs (FN). Des ces valeurs nous calculons le

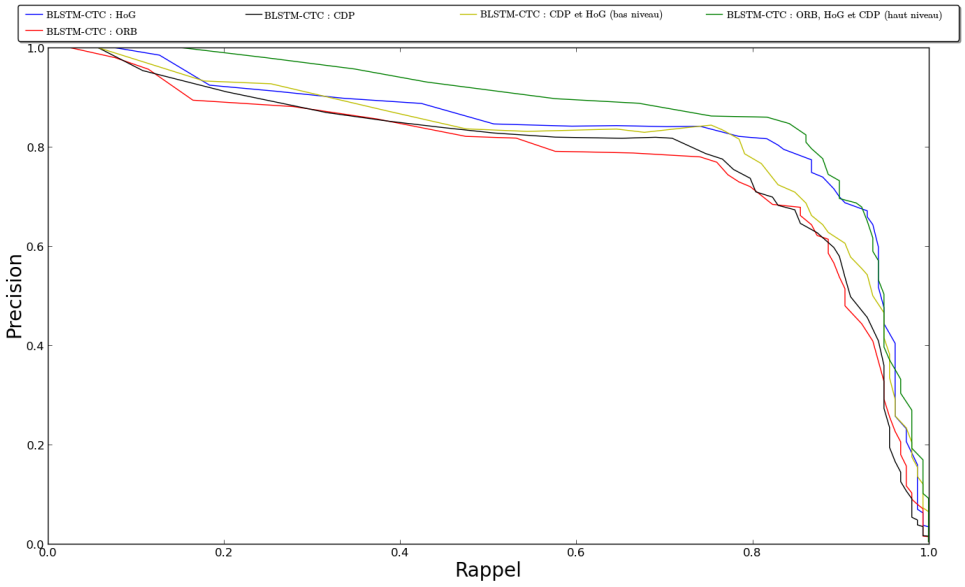


FIGURE 6.18 – Courbes rappel-précision pour les différents BLSTM-CTC

rappel R et la précision P :

$$R = \frac{VP}{VP + FN} \quad P = \frac{VP}{VP + FP} \tag{6.3}$$

ce qui nous permet de tracer des courbes de rappel-précision. Le point de fonctionnement optimal dépend de notre application, soit nous désirons utiliser un système présentant peu de lignes sans le mot clef cherché, mais ne détectant pas certaines lignes contenant des mots clefs (précision plus important que le rappel), soit nous désirons un système proposant un grand nombre d'images de lignes, avec un nombre plus important de lignes ne contenant pas le mot clef (rappel plus important que la précision).

À l'aide de la courbe de la figure 6.18, nous pouvons voir que le BLSTM-CTC le plus performant dans le cas de la requête textuelle est le BLSTM-CTC multi-caractéristiques avec combinaison de haut niveau (moyenne et BLSTM-CTC). Nous validons donc l'utilisation de ce BLSTM-CTC pour les requêtes par image. Il est cependant surprenant de voir que le BLSTM-CTC avec des caractéristiques HoG offre des performances plus intéressantes par rapport à la combinaison de bas niveau HoG et CDP et surtout que le BLSTM-CTC avec ORB ait les performances les plus faibles (pour la précision et le rappel). Le BLSTM-CTC avec ORB produit à la fois plus de faux positifs et plus de faux négatifs que le BLSTM-CTC avec CDP ou HoG, les erreurs qu'il commet doivent donc être très importantes pour empêcher le MMC de s'aligner

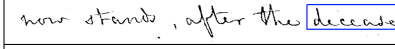
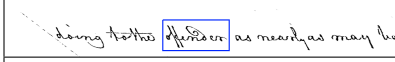
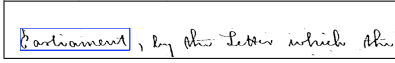
Image prétraitée	Mot clef	ORB	HoG
	decease	de cease	diceare
	offender	ofender	offender
	parliament	sartiavement	parliament

TABLE 6.5 – Comparaison des sorties textes du corpus avec HoG et ORB

sur la séquence. En observant les sorties textes (en utilisant le Top 1 des probabilités a posteriori) du corpus ligne nous obtenons des erreurs présentées dans le tableau 6.5. Depuis ce tableau nous pouvons voir que les sorties textes des BLSTM-CTC sur les zones correspondantes aux mots clefs recherchés que les caractéristiques ORB génèrent beaucoup d'ajouts ou de suppressions de caractères qui sont mal gérées par notre MMC.

Les 90 requêtes images contiennent des images des mots des requêtes textuelles, comme montré dans la figure 6.19.

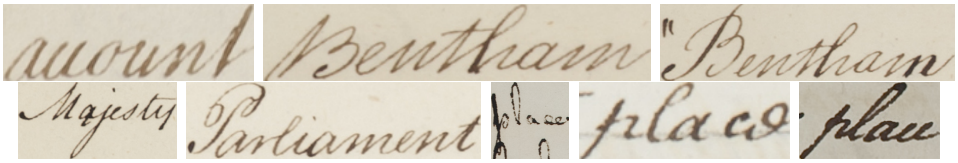


FIGURE 6.19 – Exemples de requêtes images

Les courbes rappel-précision de la figure 6.20 présente une comparaison des résultats obtenus entre les trois systèmes de détection de mots clefs.

La courbe bleue représente le système de requêtes images, la courbe rouge le système de requêtes images avec lexique et la courbe verte le système de requêtes textuelles. Le système de requêtes images sans appui linguistique offre des performances très faible comparativement aux deux autres, ce système cumule les erreurs de la reconnaissance du corpus ainsi que de la reconnaissance de l'image requête. Les erreurs se cumulent, mais ne se compensent pas, ce qui montre bien la variabilité de l'écriture manuscrite dans cette base. Les erreurs que nous avons sont liées à la reconnaissance effectuée par le BLSTM-CTC, par exemple l'image montré en figure 6.21a est identifiée par le BLSTM-CTC comme « rcount ». Ce résultat peut être expliqué par l'apparence de l'image une fois prétraitée (voir la figure 6.21b) qui peut aussi être lu par un humain comme « rcount ».

D'autres résultats ne peuvent pas être expliqué par une dégradation effectuée par les prétraitements, mais proviennent sans doute d'un mauvais ap-

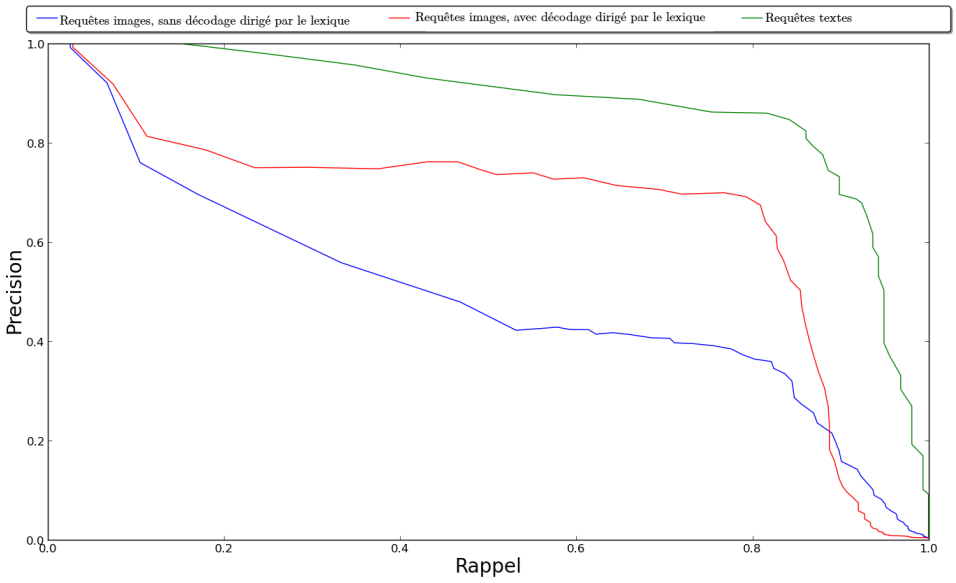


FIGURE 6.20 – Courbes rappel-précision pour les différents systèmes

(a) Image originale (b) Image prétraitée

FIGURE 6.21 – Images du mot « amount »

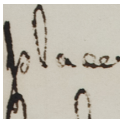
prentissage des BLSTM-CTC. L'image du mot « Majesty »de la figure 6.22a produit le mot « Masesty » alors que l'image prétraitée est très lisible (voir figure 6.22b).

(a) Image originale (b) Image prétraitée

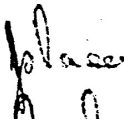
FIGURE 6.22 – Images du mot « Majesty »

Le système de requêtes images avec correction par le lexique est une amélioration importante du système précédent. Il permet de corriger des erreurs simples comme pour l'image de la figure 6.21. D'autres ne sont pas corrigées, car le mot proposé par le BLSTM-CTC est présent dans le lexique. C'est le cas pour l'image de la figure 6.23 qui est reconnu par le BLSTM-CTC comme « stage ».

Néanmoins les systèmes de requêtes images restent bien moins performant



(a) Image originale

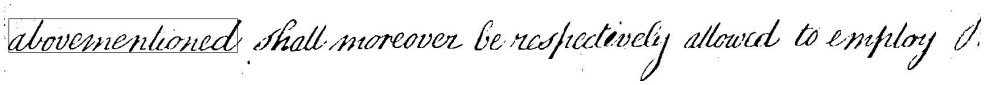


(b) Image prétraitée

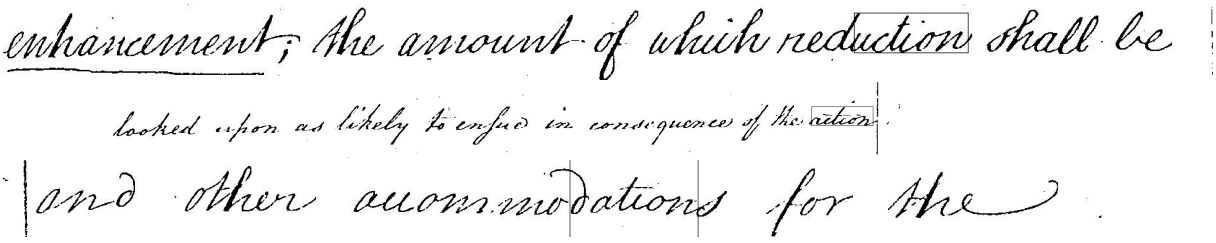
FIGURE 6.23 – Images du mot « place »

qu'un système de requêtes textuelles, un simple lexique n'est pas suffisant pour améliorer ces systèmes.

Au moment où nous avons écrit ce manuscrit nous n'avons pas les résultats de la compétition ICDAR 2015, nous ne pouvons donc pas nous comparer à d'autres systèmes. La figure 6.24 présente quelques exemples de nos résultats sur la base de test pour cette compétition, les zones entourées par un rectangle noir représentent les zones dans lesquelles nous avons détecté le mot clef.



(a) Mot clef « abovementioned »



(b) Mot clef « action »

FIGURE 6.24 – Exemples de détection de mots clefs dans des images

À titre de comparaison pour les requêtes image, l'article de I. Pratikakis, K. Zagoris, B. Gatos, et al. [142] présente des résultats produits sur une autre partie de la base Bentham, avec d'autres requêtes, lors de ICFHR 2014. En nous comparant aux courbes présentes dans cet article nous pouvons voir que nos systèmes avec des requêtes images offrent des performances similaires voire supérieures.

6.2.2.3 Conclusion

Dans cette section nous avons présenté trois systèmes permettant d'effectuer la détection de mots clefs. Nos systèmes de détection de mots clefs par des requêtes images sont atypiques car ils ne procèdent pas par une recherche d'images similaires comme c'est le cas dans la littérature [6, 142]. Nous transformons ces requêtes images en requêtes textuelles à l'aide d'un classifieur dyna-

mique BLSTM-CTC, dans l'un des systèmes le résultat texte du BLSTM-CTC n'est pas corrigé, dans l'autre système il est corrigé par un MMC effectuant un décodage dirigé par le lexique. La requête textuelle est ensuite traitée par un MMC avec un décodage Viterbi qui permet d'obtenir un score sur la présence des mots clefs recherchés. Le seuillage de ce score permet de décider si une image de ligne contient la requête. Parmi ces trois systèmes, la requête textuelle est la plus performante, en effet étant donné notre système les erreurs ne proviennent que d'une source : la conversion de notre corpus d'images par le BLSTM-CTC en un corpus de probabilités a posteriori des caractères. Les deux autres systèmes combinent les erreurs du corpus de probabilités et de la transformation d'une requête image vers une requête textuelle. Néanmoins il serait intéressant d'appliquer des méthodes de mesures de correspondances de chaînes de caractères inexactes [76, 88] pour mesurer l'impact que ces méthodes ont sur notre système : nous risquons à la fois d'ajouter un plus grand nombre de faux positifs (faisant ainsi baisser la précision), tout en reconnaissant un plus grand nombre de vrais positifs (augmentant ainsi le rappel et la précision).

Notre système effectuant des requêtes textuelles peut être utilisé dans le cadre d'un moteur de recherche sur des documents images. Il peut aussi être utilisé pour déterminer des documents au contenu similaire [49, 96].

Nous souhaitons soulever une interrogation concernant la recherche de mots clefs, notamment dans le cadre de la compétition ICDAR 2015. En effet nous pensons qu'il s'agit d'un problème mal posé : une requête textuelle a pour but de permettre à un utilisateur d'accéder à un ensemble de documents contenant le mot clef recherché, mais le but d'une requête par image n'est pas aussi clair. L'utilisateur désire t-il accéder à un contenu texte similaire à celui présent dans l'image ? Ou bien désire t-il accéder à une image avec une morphologie similaire (par exemple trouver un mot similaire produit par le même scripteur) ? Cette image pourrait alors contenir un mot différent, ou au contraire pourrait ne pas accepter le même mot d'un autre scripteur. Dans le cas ou nous désirons accéder à une image avec une morphologie similaire notre système n'est pas adapté. De notre point de vue le problème de la détection de mots clefs à partir d'une image est donc un problème mal posé, qu'il conviendrait de mieux définir.

6.3 Conclusion

Dans ce chapitre nous avons proposé deux applications originales de la reconnaissance de l'écriture manuscrite. Que ce soit pour la détection de mots clefs à partir de requêtes images ou pour la reconnaissance de la langue. Très peu d'applications utilisent une phase de reconnaissance de l'écriture préalable pour ces tâches. Cette intégration est cependant rendue possible par la qualité discriminative des BLSTM-CTC. En effet des systèmes antérieurs (systèmes MMC ou Neuro-Markoviens) bien qu'offrant des performances satisfaisantes ne permettaient pas d'arriver à un niveau ou le modèle d'attache aux données seul permettaient d'effectuer la reconnaissance de texte suffisamment bonne. C'est ce que l'on voit en particulier au travers des performances réalisées sur la base Bentham pour la détection de mots clefs par des requêtes textuelles.

D'autres applications de la reconnaissance de l'écriture manuscrite sont possibles : détection d'entité nommés dans du texte[102, 112, 137], découverte de documents avec du contenu similaire. La reconnaissance de l'écriture ne se limite pas à une simple numérisation au format texte d'une image de document, elle permet d'ouvrir aux images de documents l'ensemble des traitements déjà existants pour les documents au format texte. Cependant comme nous l'avons vu, nous ne sommes pas encore à des seuils où les systèmes de reconnaissance de l'écriture permettent de remplacer un opérateur humain. Bien qu'imparfait nous, humains, sommes capables de faire appel à une grande quantité de mémoire et de savoir en un laps de temps très court pour prendre une décision. Par exemple nous pouvons faire usage du contexte d'une phrase dans un document, ou dans une situation donnée pour déterminer le sens d'un texte et déterminer le contenu texte d'une image difficilement lisible. Les chercheurs tentent de produire des algorithmes capable d'introduire des notions de contexte dans un document [32, 138], mais nous sommes encore loin de pouvoir intégrer un contexte aussi large qu'un humain.

Chapitre 7

Conclusion générale

L'analyse automatique de documents papiers numérisés est une tâche complexe qui combine différentes techniques et méthodes développées autour de la vision par ordinateur. Dans nos travaux nous nous sommes focalisés sur une sous partie de cette analyse, à savoir la reconnaissance de l'écriture manuscrite. Nous n'avons pas abordé les problématiques en amont, comme par exemple les prétraitements effectués sur une image de documents ou l'extraction des zones de textes ou encore l'extraction des lignes de texte ; ni les problématiques en aval, c'est à dire l'extraction d'une structure logique du document ou la similarité de documents. Les travaux présentés dans cette thèse ont permis de mettre en avant des avancées sur la reconnaissance de l'écriture et des applications connexes.

Dans un premier temps notre étude bibliographique a présenté les algorithmes et les méthodes statistiques de classification de séquences utilisées pour la reconnaissance de l'écriture. Nous avons présenté les différentes familles de systèmes utilisés pour la reconnaissance de l'écriture manuscrite, ce qui a permis de mettre en avant les systèmes état de l'art. Cette étude nous a permis par la suite de réaliser des systèmes de reconnaissance de l'écriture afin de résoudre différents problèmes.

Dans un second temps nous avons présenté une contribution dans laquelle nous comparons différents systèmes appartenant aux familles de systèmes présentés précédemment. Dans cette étude comparative nous avons principalement comparé différents classifieurs dynamiques et nous avons montré que même sur des bases que l'on peut qualifier de simples, la reconnaissance de l'écriture est un problème non résolu et ce malgré les améliorations importantes apportées aux classifieurs les plus performants. Nous avons aussi comparé le fonctionnement de ces classifieurs dynamiques avec différentes familles de caractéristiques. À partir de cette étude nous avons pu identifier un système très performant, basé sur des réseaux récurrents, pour lequel nous proposons différentes améliorations.

L'une de ces améliorations constitue notre seconde contribution. Nous avons

observé dans notre contribution précédente que des réseaux récurrents appris sur différentes familles de caractéristiques produisent des erreurs différentes, ils peuvent donc se complémenter pour corriger les erreurs du système. Notre contribution explore les stratégies de combinaisons de réseaux récurrents. Ces combinaisons opèrent à différents niveaux de la structure des réseaux récurrents : bas niveau (en entrée), moyen niveau (dans le réseau), haut niveau (en sortie). Nous avons montré que selon le système final utilisé différentes combinaisons sont à privilégier. Un système dont le but est d'identifier des mots appartenant à un lexique utilisera une combinaison de moyen niveau qui permet de proposer un plus grand nombre de caractères alternatifs tandis qu'un système dont le but est d'identifier des mots hors lexique fera appel à une combinaison de haut niveau permettant d'obtenir les caractères les plus probables.

Nos dernières contributions mettent en avant des applications pour lesquels nous avons eu une approche originale : la reconnaissance de la langue et la détection de mots clefs. Les systèmes de reconnaissance de la langue de la littérature extraient des caractéristiques des images de texte pour classifier directement la langue. Néanmoins cette approche n'est pas satisfaisante car des langues du même script ont un aspect graphique très similaire. Nous avons présenté un système de reconnaissance de la langue dans des images de texte manuscrit utilisant une phase de reconnaissance de l'écriture avant d'utiliser des statistiques sur la fréquence des quadri-grammes de caractères pour déterminer la langue. Cette méthode est plus pertinente car les langues d'un même script se distinguent par les quadri-grammes les plus fréquents.

Les systèmes de détection de mots clefs par des requêtes images extraient des descripteurs de la requête soumise au système par l'utilisateur, permettant par la suite de détecter des images similaires. Cette approche n'est pas satisfaisante car nous n'effectuons pas une recherche de mots clefs, mais une recherche de similarité d'images. Au contraire des approches précédentes, notre approche transforme une requête image en une requête de texte avant d'effectuer une recherche du mot clef, produit par l'étape de reconnaissance de l'écriture, sur un corpus d'images. Ces approches ont été rendues possibles par les améliorations récentes des systèmes de reconnaissance de l'écriture. En effet, les systèmes de reconnaissance de l'écriture mis en place pour ces deux applications ont un niveau d'erreurs caractères assez bas pour permettre de les utiliser associés à des connaissances linguistiques. Nous avons pu voir à partir des performances de ces applications, que nous avons mesurées sur des bases complexes, qu'il

reste encore beaucoup à faire pour permettre à des systèmes de reconnaissance de l'écriture d'atteindre des performances équivalentes à celle d'un humain.

Les perspectives de nos travaux qui apparaissent à l'issue de cette thèse peuvent être divisées en trois axes : améliorations des combinaisons, applications aux domaines connexes à la reconnaissance de l'écriture, applications à des domaines différents.

Les combinaisons de caractéristiques peuvent être améliorées, en particulier la combinaison à moyen niveau. En effet comme nous l'avons expliqué dans la section 5.1.2, la combinaison à moyen niveau combine plusieurs réseaux de neurones récurrents à l'aide d'un nouveau réseau de neurones (récurrent ou non). Ce réseau de combinaison est appris sur les sorties des réseaux de neurones précédent, seules ses pondérations sont modifiées. Cependant nous pourrions possiblement améliorer ces résultats en optimisant de nouveau les pondérations des réseaux de neurones récurrents produisant les sorties qui sont utilisées par le réseau de combinaison. Cette optimisation permettraient d'affiner les poids des réseaux de neurones sous-jacent en les optimisant par rapport à une cible au niveau mot. Il serait aussi possible de comparer de nouveaux opérateurs de combinaisons : vote à la majorité, médiane, minimum, régression logistique, ou par des classifieurs. L'exploration de nouvelles caractéristiques est une autre piste. Avec un très grand nombre de réseaux récurrents appris sur différentes familles de caractéristiques nous pourrions améliorer les résultats. Cependant cela risque d'augmenter le temps de traitement pour reconnaître une image. Il serait alors préférable de sélectionner l'ensemble de caractéristiques ayant la plus forte complémentarité et tirer profit de cette complémentarité au travers de la combinaison la plus adaptée.

L'objectif de ces combinaisons est de corriger au maximum les erreurs du système. Hors sur des bases complexes, comme la base Bentham, même si nous avons corrigé un certain nombre de ces erreurs d'autres sont restées, et certaines ont pu être rajoutées. La correction de ces erreurs reste un objectif majeur des systèmes de reconnaissance, tôt ou tard nos systèmes ne pourront plus se contenter d'utiliser le contexte local pour transcrire une image de mot ou de ligne en un texte numérique, ils devront utiliser le contexte global du document pour prendre une décision locale performante. La correction d'erreurs locales à l'aide du contexte global est déjà effectuée par des systèmes commerciaux, et nous sommes convaincus qu'il faut continuer à faire progresser ces approches. Nous même, humains, nous opérons de cette façon, lorsqu'une zone de texte est difficile à lire nous essayons de nous référer au document, ou au contexte

du document, pour inférer une correction.

De manière générale les réseaux récurrents et les systèmes de combinaisons de ces réseaux permettent de classifier des séquences. Ils peuvent donc être appliqués dans de nombreux autres domaines dans lesquels conserver une information sur le long terme est important. Par exemple dans le domaine médical ils peuvent être appliqués à la détection d'asynchronismes entre un patient et son respirateur. Dans le cas des réseaux récurrents multi-dimensionnels ont peut envisager leur utilisation pour la segmentation d'images médicales. Ils peuvent aussi être appliqués à la reconnaissance de gestes ou à la classification d'environnements routiers. Ces nouveaux réseaux récurrents restent méconnus dans des domaines qui utilisent des classifieurs similaires mais plus anciens, les approches avec des réseaux de neurones récurrents peuvent donc apporter des gains de performances et de nouveaux progrès dans ces domaines.

Chapitre 8

Publications liées

Le travail présenté dans cette thèse a fait l'objet des publications suivantes.

G. Bideault, L. Mioulet, C. Chatelain et T. Paquet(2014). A hybrid CRF/HMM approach for handwriting recognition. *International Conference on Image Analysis and Recognition.*

Abstract : In this article, we propose an original hybrid CRF-HMM system for handwriting recognition. The main idea is to benefit from both the CRF discriminative ability and the HMM modeling ability. The CRF stage is devoted to the discrimination of low level frame representations, while the HMM performs a lexicon-driven word recognition. Low level frame representations are defined using n-gram codebooks and HOG descriptors. The system is trained and tested on the public handwritten word database RIMES.

G. Bideault, L. Mioulet, C. Chatelain et T. Paquet (2015). A Hybrid BLSTM-HMM for spotting Regular Expressions. *International Conference on Pattern Recognition Applications and Methods.*

Abstract : This article concerns the spotting of regular expressions (REGEX) in handwritten documents using a hybrid model. Spotting REGEX in a document image allow to consider further extraction tasks such as document categorization or named entities extraction. Our model combines state of the art BLSTM recurrent neural network for character recognition and segmentation with a HMM model able to spot the desired sequences. Our experiments on a public handwritten database show interesting results. 1.

L. Mioulet, G. Bideault, C. Chatelain et T. Paquet(2015a). BLSTM-CTC Combination Strategies for Off-line Handwriting Recognition. *In Internation Conference on Pattern Recognition Applications and Methods.*

Abstract : In this paper we present several combination strategies using multiple BLSTM-CTC systems. Given several feature sets our aim is to determine which strategies are the most relevant to improve on an isolated word

recognition task (the WR2 task of the ICDAR 2009 competition), using a BLSTM-CTC architecture. We explore different combination levels : early integration (feature combination), mid level combination and late fusion (output combinations). Our results show that several combinations outperform single feature BLSTM- CTCs.

L. Mioulet, G. Bideault, C. Chatelain, T. Paquet et S. Brunessaux(2015b). Exploring multiple feature combination strategies with a recurrent neural network architecture for off-line handwriting recognition. *In Document Recognition and Retrieval*.

Abstract : The BLSTM-CTC is a novel recurrent neural network architecture that has outperformed previous state of the art algorithms in tasks such as speech recognition or handwriting recognition. It has the ability to process long term dependencies in temporal signals in order to label unsegmented data. This paper describes different ways of combining features using a BLSTM-CTC architecture. Not only do we explore the low level combination (feature space combination) but we also explore high level combination (decoding combination) and mid-level (internal system representation combination). The results are compared on the RIMES word database. Our results show that the low level combination works best, thanks to the powerful data modeling of the LSTM neurons.

Les publications suivantes ont été soumises pour la conférence ICDAR 2015 mais n'ont toujours pas été validées.

L. Mioulet, U. Garain, C. Chatelain, P. Barlas et T. Paquet(2015). Language identification from handwritten documents. *In International Conference on Document Analysis and Recognition*.

Abstract : This paper presents a novel approach for language identification in handwritten documents. The approach is based on script identification followed by character recognition. BLSTM-CTC based handwriting recognizers are used and the OCR output is fed to a statistical language identifier for detecting the language of the input handwritten document. Documents in two scripts (Latin and Bengali) and four languages (English, French, Bengali and Assamese) are considered for evaluation. Several alternative frameworks have been explored, effects of handwriting recognition and text length on language detection have been studied. It is observed that with some empirical restric-

tions it is very much possible to achieve more that 80% language detection accuracy and based on the current research practical systems can be designed.

U. Garain, L. Mioulet, C. Chatelain et T. Paquet(2015). Unconstrained Bengali Handwriting Recognition with Recurrent Models. *In International Conference on Document Analysis and Recognition*.

Abstract : This paper presents a pioneering attempt for developing a recurrent neural net based connectionist system for unconstrained Bengali offline handwriting recognition.The major challenge in configuring a such classification system for a script like Bengali is to effectively define the character classes. A novel way of defining character classes is introduced in order to make the recognition problem suitable for using a recurrent model which has to deal with more than nine hundred character classes for which the occurrence probability is very skewed in the language. An off-the-shelf BLSTM-CTC recognizer is used. A new open-source dataset is developed for unconstrained Bengali offline handwriting recognition. The dataset contains about 2000 handwritten text lines consisting of about 18,000 words. Experiment shows that with the new definition of character classes the BLSTM-CTC framework provides an impressive performance for unconstrained Bengali offline handwriting recognition. The character level recognition accuracy is 75.40% without doing any post-processing on the BLSTM-CTC output. Among the 24.60% character level errors, the substitution, deletion and insertion errors are 18.91%, 4.69% and 0.98%, respectively.

Bibliographie

Abirami, S. and Manjula, D. (2009). A Survey of Script Identification techniques for Multi-Script Document Images. *International Journal of Recent Trends in Engineering*, 1(2) :246–249.

Ahmad, A., Viard-Gaudin, C., Khalid, M., and Yusof, R. (2004). Online handwriting recognition using support vector machine. *IEEE Region 10 Conference TENCON*, pages 311–314.

Ait-Mohand, K., Paquet, T., and Ragot, N. (2014). Combining structure and parameter adaptation of HMMs for printed text recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (99).

Al-Hajj Mohamad, R., Likforman-Sulem, L., and Mokbel, C. (2007). Combination of HMM-based classifiers for the recognition of Arabic handwritten words. In *International Conference on Document Analysis and Recognition*, volume 2.

Al-Hajj Mohamad, R., Likforman-Sulem, L., and Mokbel, C. (2009). Combining slanted-frame classifiers for improved HMM-based arabic handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(7) :1165–1177.

Almazan, J., Gordo, A., Fornes, A., and Valveny, E. (2013). Handwritten Word Spotting with Corrected Attributes. *IEEE International Conference on Computer Vision*, pages 1017–1024.

Andrew, Y. and Jordan, M. (2002). On discriminative vs. generative classifiers : A comparison of logistic regression and naive bayes. *Advances in neural information processing systems*, 2 :841–848.

Baldi, P. and Chauvin, Y. (1994). Smooth On-Line Learning Algorithms for Hidden Markov Models. *Neural Computation*, 6(2) :307–318.

Baum, L., Petrie, T., Soules, G., and Weiss, N. (1970). A maximization Technique Occurring in the statistical analysis of probabilistic functions of Markov Chains. *The Annals of Mathematical Statistics*, 41(1) :164–171.

Bay, H., Tuytelaars, T., and Gool, L. V. (2008). SURF : Speeded Up Robust Features. *Computer Vision and Image Understanding*, 110(3) :346–359.

Becker, S. and Le Cun, Y. (1988). Improving the convergence of back-propagation learning with second order methods. In *Proceedings of the connectionist models summer school*, pages 29–37.

Bengio, Y. (2009). Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning*, 2(1) :1–127.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation Learning : A Review and New Perspectives. *IEEE transactions on pattern analysis and machine intelligence*, pages 1–34.

Bengio, Y., De Mori, R., Flammia, G., and Kompe, R. (1992). Global optimization of a neural network-hidden Markov model hybrid. *IEEE Transactions on Neural Networks*, 3(2) :252–259.

Bengio, Y. and Le Cun, Y. (2007). Scaling learning algorithms towards AI. *Large-Scale Kernel Machines*, 34 :1–41.

Bengio, Y. and LeCun, Y. (2007). Scaling learning algorithms towards AI. *Large-Scale Kernel Machines*, 34 :1–41.

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

Bianne-Bernard, A., Menasri, F., Al-Hajj Mohamad, R., Mokbel, C., Kermorvant, C., and Likforman-Sulem, L. (2011). Dynamic and Contextual Information in HMM Modeling for Handwritten Word Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(10) :2066–2080.

Bideault, G., Mioulet, L., Chatelain, C., and Paquet, T. (2015). A Hybrid BLSTM-HMM for spotting Regular Expressions. In *International Conference on Pattern Recognition Applications and Methods*.

Bideault, G., Mioulet, L., Clément, C., and Paquet, T. (2014). A hybrid CRF/HMM approach for handwriting recognition. In *International Conference on Image Analysis and Recognition*, pages 403–410.

Bilmes, J. A. (1998). A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. *International Computer Science Institute*, 4(510) :126.

Bin, Z., Srihari, S., and Huang, C. (2004). Word image retrieval using binary features. *Elecronic Imaging*.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford University Press.

Bluche, T., Louradour, J., Knibbe, M., Moysset, B., Benzeghiba, M. F., and Kermorvant, C. (2014). The A2iA Arabic Handwritten Text Recognition System at the OpenHaRT2013 Evaluation. In *International Workshop on Document Analysis Systems on Document Analysis Systems*, pages 161–165.

Bortolozzi, F., de Souza Britto Jr, A., Oliveira, L. S., and Morita, M. (2005). Recent advances in handwriting recognition. In *Proceedings of the International Workshop on Document Analysis*, pages 1–30. Citeseer.

Bosch, A., Zisserman, A., and Mu, X. (2006). Scene Classification Via pLSA. In *European Conference for Computer Vision*, pages 517–530. Springer.

Bottou, L. (2014). From machine learning to machine reasoning. *Machine Learning*.

Bourlard, H. and Morgan, N. (1991). Merging multilayer perceptrons and hidden Markov models : Some experiments in continuous speech recognition. *Neural Networks : Advances and Applications*, 3 :215–239.

Bourlard, H. and Wellekens, C. (1990). Links between Markov models and multilayer perceptrons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12 :1167–1178.

Boyer, R. and Moore, J. (1977). A fast string searching algorithm. *Communications of the ACM*, 20(10) :762–772.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1) :5–32.

Broda, B. and Piasecki, M. (2007). Correction of Medical Handwriting OCR Based on Semantic Similarity. *International conference on Intelligent data engineering and automated learning*, (3) :437–446.

Bulacu, M. and Schomaker, L. (2006). Combining Multiple Features for Text-Independent Writer Identification and Verification. In *International Workshop on Frontiers in Handwriting Recognition*, pages 281–286.

Bunke, H., Bengio, S., and Vinciarelli, A. (2004). Offline recognition of unconstrained handwritten texts using HMMs and statistical language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6) :709–720.

Bunke, H., Roth, M., and Schukat-Talamazzini, E. (1994). Off-line cursive handwriting recognition using hidden Markov models. *Pattern recognition*, 2 :383–386.

Cai, J. and Liu, Z. (1999). Integration of structural and statistical information for unconstrained handwritten numeral recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(3) :263–270.

Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). BRIEF : Binary Robust Independent Elementary Features. *European Conference for Computer Vision*, pages 778–792.

Casey, R. G. and Lecolinet, E. (1996). A Survey of Methods and Strategies in Character Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(7) :690–706.

Cayton, L. and Diego, S. (2008). Fast Nearest Neighbor Retrieval for Bregman Divergences. In *Proceedings of the 25th international conference on Machine learning*, volume 1, pages 112–129.

Cesar, M. and Shinghal, R. (1990). An algorithm for segmenting handwritten postal codes. *International Journal of Man-Machine Studies*, 33 :63–80.

Chen, G., Parada, C., and Sainath, T. (2015). Query-by-example keyword spotting using Long Short Term Memory Networks. *International Conference on Acoustics, Speech, and Signal Processing*, pages 1–5.

Clergeau-Tournemire, S. and Plamondon, R. (1995). Integration of lexical and syntactical knowledge in a handwriting-recognition system. *Machine Learning*, 8(4) :249–259.

Coates, A. and Ng, A. (2012). Learning Feature Representations with K-means. In *Neural Networks : Tricks of the Trade*, pages 561–580.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuska, P. (2011). Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12 :2493–2537.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3) :273–297.

Cruz, R., Cavalcanti, G., and Ren, T. (2010). Handwritten digit recognition using multiple feature extraction techniques and classifier ensemble. *International Conference on Systems, Signals and Image Processing*, pages 215–218.

Dahl, G., Yu, D., Deng, L., and Acero, A. (2012). Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, pages 1–13.

Dalal, N. and Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1 :886–893.

Damashek, M. (1995). Gauging Similarity with n-Grams : Language-Independent Categorization of Text. *Science*, 267(5199) :843–848.

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum Likelihood from incomplete Data via the EM algorithm. *Journal of the Royal statistical Society*, 39(1) :1–38.

Déniz, O., Bueno, G., Salido, J., and De La Torre, F. (2011). Face recognition using Histograms of Oriented Gradients. *Pattern Recognition Letters*, 32(12) :1598–1603.

Dewan, S. and Chakravarthy, S. (2012). A system for offline character recognition using auto-encoder networks. *Neural Information Processing*, pages 91–99.

DGA, AirbusDS, and LNE. MAURDOR : Moyens Automatisés pour la Reconnaissance de Documents Écrits.

Dharanipragada, S. and Roukos, S. (1998). A fast vocabulary independent algorithm for spotting words in speech. *International Conference on Acoustics, Speech, and Signal Processing*, 1 :233–236.

Doermann, D. (1998). The Indexing and Retrieval of Document Images : A Survey. *Computer Vision and Image Understanding*, 70(3) :287–298.

Dong, J., Ponson, D., Suen, C., and Krzyzak, A. (2005). Cursive word skew/slant corrections based on Radon transform. *Proceedings of the 8th International Conference on Document Analysis and Recognition*, pages 478–483.

Dreuw, P., Doetsch, P., Plahl, C., and Ney, H. (2009). Hierarchical Hybrid MLP/HMM or rather MLP Features for a Discriminatively Trained Gaussian HMM : a Comparison for Offline Handwriting Recognition. *International Conference on Image Processing*, 41 :1–5.

Dunn, C. E. and Wang, P. S. P. (1992). Character segmentation techniques for handwritten text-a survey. *International Conference on Pattern Recognition Methodology and Systems*, pages 577–580.

Dunning, T. (1994). *Statistical Identification of Language*. Computing Research Laboratory, New Mexico State University.

El-Yacoubi, A., Bertille, J., and Gilloux, M. (1995). Conjoined location and recognition of street names within a postal address delivery line. *Proceedings of the Third International Conference on Document Analysis and Recognition*, 2 :1024–1027.

El-Yacoubi, A., Gilloux, M., Sabourin, R., and Suen, C. Y. (1999a). An HMM-based approach for off-line unconstrained handwritten word modeling and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8) :752–760.

El-Yacoubi, A., Sabourin, R., Gilloux, M., and handwritten word recognition using hidden markov models Suen, C.-l. (1999b). Off-Line Handwritten Word Recognition using Hidden Markov Models. *Knowledge-based intelligent techniques in character recognition*, pages 193–229.

Erhan, D., Manzagol, P. A., Bengio, Y., Bengio, S., and Vincent, P. (2009). The difficulty of training deep architectures and the effect of unsupervised pre-training. *Proceedings of The Twelfth International Conference on Artificial Intelligence and Statistics*, 5 :153–160.

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

Fink, G., Hammerla, N., Plötz, T., and Vajda, S. (2011). Towards Feature Learning for HMM-based Offline Handwriting Recognition.

Fischer, A., Keller, A., Frinken, V., and Bunke, H. (2010). HMM-based word spotting in handwritten documents using subword models. *International Conference on Pattern Recognition*, pages 3416–3419.

Fischer, A., Keller, A., Frinken, V., and Bunke, H. (2012). Lexicon-free handwritten word spotting using character HMMs. *Pattern Recognition Letters*, 33(7) :934–942.

Foote, J. T., Jones, G., Jones, K., and Young, S. (1995). Talker-Independent Keyword Spotting for Information Retrieval. *Eurospeech*, 95 :1–4.

Forney, D. (1973). The Viterbi Algorithm. *Proceedings of the IEEE*, 61(3) :302–309.

Freeman, H. (1961). On the encoding of arbitrary geometric configurations. *IRE Transactions on Electronic Computers*, pages 260–268.

Frinken, V., Fischer, A., Manmatha, R., and Bunke, H. (2012). A Novel Word

Spotting Method Based on Recurrent Neural Networks. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–14.

Ganapathiraju, A. and Picone, J. (2000). Hybrid SVM/HMM architectures for speech recognition. *INTERSPEECH*, pages 504–507.

Gehler, P. and Nowozin, S. (2009). On feature combination for multiclass object classification. In *IEEE International Conference on Computer Vision*, pages 221–228. IEEE.

Gers, F. A. and Schraudolph, N. N. (2002). Learning Precise Timing with LSTM Recurrent Networks. *Journal of Machine Learning Research*, 3 :115–143.

Glauberman, M. (1956). Character Recognition for business machines. In *Electronics*, pages 132–136.

Gonzalo, N. (2001). A guided tour to approximate sring matching. *ACM Computing surveys*, 33(1) :31–88.

Graves, A. (2008). *Supervised sequence labelling with recurrent neural networks.* PhD thesis.

Graves, A. (2011). RNNLIB : A recurrent neural network library for sequence learning problems.

Graves, A., Fernández, S., and Schmidhuber, J. (2007). An application of recurrent neural networks to discriminative keyword spotting. *International Conference on Artificial Neural Networks*, pages 220–229.

Graves, A. and Gomez, F. (2006). Connectionist temporal classification : Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning.*

Graves, A., Liwicki, M., Bunke, H., Santiago, F., and Schmidhuber, J. (2008). Unconstrained on-line handwriting recognition with recurrent neural networks. *Advances in Neural Information Processing Systems*, 20 :1–8.

Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., and Schmidhuber, J. (2009). A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5) :855–68.

Graves, A. and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural networks : the official journal of the International Neural Network Society*, 18(5-6) :602–10.

Graves, A. and Schmidhuber, J. (2008). Offline handwriting recognition with multidimensional recurrent neural networks. *Advances in Neural Information Processing Systems*, 21 :545–552.

Grosicki, E. and Abed, H. E. (2009). ICDAR 2009 Handwriting Recognition Competition. In *10th Interational Conference on Document Analysis and Recognition*, pages 1398–1402.

Gupta, M. and Kakde, B. (2013). A Novel Approach for Cursive Handwriting Detection Using Artificial Neural Network. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(11) :674–680.

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3 :1157–1182.

Hall, P. and Geoff, D. (1980). Approximate string matching. *ACM Computing surveys1*, 12(4) :381–402.

Hamdani, M., Doetsch, P., Kozielski, M., Mousa, A., and Ney, H. (2013). The RWTH Large Vocabulary Arabic Handwriting Recognition System. In *NIST Open Handwriting and Recognition Workshop*.

Hamdani, M., Doetsch, P., and Ney, H. (2014). Improvement of Context Dependent Modeling For Arabic Handwriting Recognition. In *International Conference on Frontiers in handwriting recognition*.

Heutte, L., Paquet, T., Moreau, J., Lecourtier, Y., and Olivier, C. (1998). A structural/statistical feature based vector for handwritten character recognition. *Pattern Recognition Letters*, 19(7) :629–641.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786) :504–7.

Hochberg, J., Bowers, K., Cannon, M., and Patrick, K. (1999). Script and Language Identification for Handwritten Document Images. *International Journal Document Analysis and Recognition*, 2(2/3) :45–52.

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

Hoffman, R. and McCullough, J. (1971). Segmentation methods for recognition of machine-printed characters. *IBM Journal of Research and Development*, 15(2) :153–165.

Hofmann, T. (2000). Learning the Similarity of Documents. *Advances in Neural Information Processing Systems*, pages 914–920.

Hu, R. and Collomosse, J. (2013). A Performance Evaluation of Gradient Field HOG Descriptor for Sketch Based Image Retrieval. *Computer Vision and Image Understanding*, 117(7) :790–806.

Hume, A. and Sunday, D. (1991). Fast string searching. *Software Practice and Experience*, 21(11) :1221–1248.

Jaitly, N. and Hinton, G. E. (2013). Using an autoencoder with deformable templates to discover features for automated speech recognition. pages 2–5.

Jolliffe, I. (2005). *Principal Component Analysis, Second Edition*.

Keaton, P., Greenspan, H., and Goodman, R. (1997). Keyword spotting for cursive document retrieval. *Proceedings Workshop on Document Image Analysis*.

Kimura, F., Miyake, Y., and Shridhar, M. (1995). Handwritten ZIP code recognition using lexicon free word recognition algorithm. *International Conference on Document Analysis and Recognition*, 2 :906–910.

Kittler, J., Hater, M., and Duin, R. P. W. (1996). Combining classifiers. In *Proceedings of the International Conference on Pattern Recognition*, volume 2, pages 897–901.

Knerr, S., Anisimov, V., Barret, O., Gorski, N., Price, D., and Simon, J. (1997). The A2iA intercheque system : courtesy amount and legal amount recognition for French checks. *International journal of pattern recognition and artificial intelligence*, 11(4) :505–548.

Koerich, A. L., Britto, A., de Oliveira, L., and Sabourin, R. (2006). Fusing High- and Low-Level Features for Handwritten Word Recognition. *International Conference on Frontiers in handwriting recognition*.

Konidaris, T., Gatos, B., Ntzios, K., Pratikakis, I., Theodoridis, S., and Perantonis, S. J. (2007). Keyword-guided word spotting in historical printed documents using synthetic data and user feedback. *International Journal on Document Analysis and Recognition*, 9(2-4) :167–177.

Kuncheva, L. and Whitaker, C. (2003). Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine learning*, 51 :181–207.

Kuncheva, L. I. (2002). A theoretical study on six classifier fusion strategies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2) :281–286.

Kundu, A., He, Y., and Bahl, P. (1988). Recognition of handwritten word : first and second order hidden Markov model based approach. *Computer Vision and Pattern Recognition*, 22(3) :457–462.

Kundu, A., He, Y., and Chen, M. (1998). Alternatives to variable duration HMM in handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11) :1275–1280.

Lafferty, J., Mccallum, A., and Pereira, F. (2001). Conditional Random Fields : Probabilistic Models for Segmenting and Labeling Sequence Data. *International Conference on Machine Learning*, pages 282–289.

LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., and Jackel, L. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1 :541–551.

LeCun, Y., Bottou, L., and Orr, G. (1998). Efficient backprop. In *Neural networks : Tricks of the trade*, pages 9–50.

Lee, D., Nohl, C., and Baird, H. (1998). Language Identification in Complex, Unoriented and Degraded Document Images. *Series in Machine Perception and Artificial Intelligence*, pages 17–39.

Leydier, Y., Lebourgeois, F., and Emptoz, H. (2007). Text search for medieval manuscript images. *Pattern Recognition*, 40(12) :3552–3567.

Liu, C. and Suen, C. (2009). A new benchmark on the recognition of handwritten Bangla and Farsi numeral characters. *Pattern Recognition*, 42(12) :3287–3295.

Liu, Y., Lin, C., and Chang, F. (2005). Language identification of character images using machine learning techniques. *International Conference on Document Analysis and Recognition*, 2 :630–634.

Louradour, J. and Kermorvant, C. (2013). Curriculum Learning for Handwritten Text Line Recognition. *arXiv preprint arXiv :1312.1737*, pages 1–9.

Lowe, D. (1999). Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, pages 1150–1157. Ieee.

Lu, S. and Tan, C. (2008). Script and language identification in noisy and degraded document images. *Transactions on Pattern Analysis and Machine Intelligence*, 30(1) :14–24.

Ma, H. and Doermann, D. (2004). Word Level Script Identification for Scanned Document Images. *SPIE Conference on Document Recognition and Retrieval*, pages 124–135.

Madhvanath, S., Kim, G., and Govindaraju, V. (1999). Chaincode Contour Processing for Handwritten Word Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9) :928–932.

Mandic, D. and Chambers, J. (2000). A normalised real time recurrent learning algorithm. *Signal processing*, 80(9) :1909–1916.

Martens, J. (2010). Deep learning via Hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, volume 951, pages 1–8.

McCallum, A. (2002). Efficiently inducing features of conditional random fields. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pages 403–410.

McCulloch, W. and Walter, P. (1943). A Logical Calculus of Ideas Immanent in Nervous Activity. 5 :115–133.

Menasri, F., Louradour, J., Bianne-Bernard, A., and Kermorvant, C. (2012). The A2iA French handwriting recognition system at the Rimes-ICDAR2011 competition. *Society of Photo-Optical Instrumentation Engineers*, 8297 :51.

Mikolajczyk, K. and Schmid, C. (2005). Performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10) :1615–30.

Mioulet, L., Bideault, G., Chatelain, C., and Paquet, T. (2015a). BLSTM-CTC Combination Strategies for Off-line Handwriting Recognition. In *Internation Conference on Pattern Recognition Applications and Methods*.

Mioulet, L., Bideault, G., Chatelain, C., Paquet, T., and Brunessaux, S. (2015b). Exploring multiple feature combination strategies with a recurrent neural network architecture for off-line handwriting recognition. In *Document Recognition and Retrieval*.

Mioulet, L., Breckon, T. P., Mouton, A., Liang, H., and Morie, T. (2013). Gabor Features for Real Time Road Environment Classification. In *Proceeding of the IEEE International Conference on Industrial Technology*, number Section III.

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

Morency, L., Quattoni, A., and Darrell, T. (2007). Latent-Dynamic Discriminative Models for Continuous Gesture Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

Morgan, N. and Bourlard, H. (1990). Continuous speech recognition using multilayer perceptrons with hidden Markov models. *Processing of the International Conference on Acoustics, Speech, and Signal*, 413-416.

Mori, S., Suen, C. Y., and Yamamoto, K. (1992). Historical review of OCR research and development. *Proceedings of the IEEE*, 7 :1029–1058.

Morita, M., Bortolozzi, F., Facon, J., Garnés, S., and Sabourin, R. (1999). Mathematical morphology and weighted least squares to correct handwriting baseline skew. *International Conference on Document Analysis and Recognition*, pages 430–434.

Nadeau, D. and Satoshi, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, pages 3–26.

Nagata, M. (1996). Context-based spelling correction for Japanese OCR. *International conference on Computational linguistics*, 2 :806.

Okazaki, N. (2007). CRFsuite : a fast implementation of Conditional Random Fields (CRFs).

Otsu, N. (1975). A threshold selection method from gray-level histograms. *Automatica*, 11(1) :23–27.

Pesch, H. and Hamdani, M. (2012). Analysis of Preprocessing Techniques for Latin Handwriting Recognition. *International Conference on Frontiers in Handwriting Recognition*, pages 280–284.

Pratikakis, I., Zagoris, K., Gatos, B., Louloudis, G., and Stamatopoulos, N. (2014). ICFHR 2014 Competition on Handwritten Keyword Spotting (H-KWS 2014). *International Conference on Frontiers in Handwriting Recognition*, pages 814–819.

Quattoni, A., Wang, S., Morency, L., Collins, M., Darrell, T., and Csail, M. (2007). Hidden-state Conditional Random Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(10) :1–17.

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

Rabiner, L. and Juang, B. (1986). An introduction to hidden Markov models. *ASSP Magazine, IEEE*, Appendix 3(January) :Appendix 3A.

Rath, T. and Manmatha, R. (2007). Word spotting for historical documents. *International Journal on Document Analysis and Recognition*, 9(2-4) :139–152.

Reynolds, D. A. and Rose, R. C. (1995). Robust text-independent speaker identification using Gaussian mixture speaker models.

Riedmiller, M. and Braun, H. (1993). A direct adaptive method for faster backpropagation learning : The RPROP algorithm. In *IEEE International Conference on Neural Networks*, pages 586–591.

Robinson, T., Hochberg, M., and Renals, S. (1996). The use of recurrent neural networks in continuous speech recognition. *Automatic speech and speaker recognition*, pages 233–258.

Rosenblatt, F. (1958). The perceptron : A probabilistic model for information storage and organization in the brain. *Psychological review*.

Rosenfeld, R. (1996). A maximum entropy approach to adaptive statistical language modelling. *Computer Speech & Language*, 10 :187–228.

Rothacker, L. (2011). *Learning bag-of-features representations for handwriting recognition*. PhD thesis.

Rublee, E. and Rabaud, V. (2011). ORB : an efficient alternative to SIFT or SURF. In *IEEE International Conference on Computer Vision*, number 2011, pages 2564–2571.

Saldarriaga, S., Morin, E., and Viard-Gaudin, C. (2009). Impact de la reconnaissance de l'écriture en-ligne sur une tâche de catégorisation. In *Conférence en Recherche d'Information et Applications*, pages 219–234.

Sarkar, R., Das, N., Basu, S., Kundu, M., Nasipuri, M., and Basu, D. (2012). CMATERdb1 : a database of unconstrained handwritten Bangla and Bangla-English mixed script document image. *International Journal on Document Analysis and Recognition*, 15(1) :71–83.

Sato, M. (1990). A real time learning algorithm for recurrent analog neural networks. *Biological Cybernetics*, 62(3) :237–241.

Sauvola, J. and Pietika, M. (2000). Adaptive document image binarization. *Pattern Recognition*, 33(2) :225–236.

Schmidhuber, J., Gers, F., and Eck, D. (2002). Learning nonregular languages : a comparison of simple recurrent networks and LSTM. *Neural computation*, 14(9) :2039–41.

Schomaker, L. and Segers, E. (1999). Finding features used in the human reading of cursive handwriting. *International Journal on Document Analysis and Recognition*, 2(1) :13–18.

Schulz, K. U. and Mihov, S. (2002). Fast string correction with Levenshtein automata. *International Journal on Document Analysis and Recognition*, 5(1) :67–85.

Schuster, M. and Paliwal, K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11) :2673–2681.

Seide, F., Li, G., Chen, X., and Yu, D. (2011). Feature engineering in context-dependent deep neural networks for conversational speech transcription. *IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 24–29.

Sekita, I., Toraichi, K., Ryoichi, M., Yamamoto, K., and Yamada, H. (1988). Feature extraction of handwritten japanese characters by spline functions for relaxation matching. *Pattern Recognition*, 21(1) :9–17.

Senior, A. and Robinson, A. (1998). An off-line cursive handwriting recognition system. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3) :309–311.

Sha, F. and Pereira, F. (2003). Shallow Parsing with Conditional Random Fields. In *Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, number 1, pages 1071–1079.

Simon, G. and Bunke, H. (2004). HMM-based handwritten word recognition : on the optimization of the number of states, training iterations and Gaussian components. *Pattern Recognition*, 37(10) :2069–2079.

Sites, D. Compact Language Detector 2.

Spitz, A. (1997). Determination of the script and language content of document images. *Transactions on Pattern Analysis and Machine Intelligence*, 19(3) :235–245.

Sutton, C., McCallum, A., and Rohanimanesh, K. (2007). Dynamic conditional random fields : Factorized probabilistic models for labeling and segmenting sequence data. *The Journal of Machine Learning Research*, 8 :693–723.

Swietojanski, P., Ghoshal, A., and Renals, S. (2013). Revisiting hybrid and GMM-HMM system combination techniques. *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6744–6748.

Tagougui, N., Boubaker, H., Kherallah, M., and Alimi, A. M. (2014). A Hybrid NN/HMM Modeling Technique for Online Arabic Handwriting Recognition. *arXiv preprint arXiv :1401.0486*.

Tan, G., Viard-Gaudin, C., and Kot, A. (2009). Information Retrieval Model for Online Handwritten Script Identification. *International Conference on Document Analysis and Recognition*.

Tappert, C., Suen, C., and Wakahara, T. (1990). The state of the art in online handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(8) :787–808.

Tauschek (1935). Reading Machine.

Thomas, S., Chatelain, C., Heutte, L., and Paquet, T. (2010). An information extraction model for unconstrained handwritten documents. *Proceedings - International Conference on Pattern Recognition*, pages 3412–3415.

Thomas, S., Chatelain, C., Heutte, L., and Paquet, T. (2013). Un modèle neuro markovien profond pour l'extraction de séquences dans des documents manuscrits. *revue Documents Numériques*, 16(2) :49–69.

Thomas, S., Chatelain, C., Heutte, L., Paquet, T., and Kessentini, Y. (2015). A Deep HMM model for multiple keywords spotting in handwritten documents. *Pattern Analysis & Applications*.

Trier, O., Jain, A., and Taxt, T. (1996). Feature extraction methods for character recognition – a survey. *Pattern recognition*.

Trier, O. and Taxt, T. (1995). Evaluation of binarization methods for document images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(3) :312–315.

Vincent, P., Larochelle, H., Yoshua, B., and Manzagol, P. A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the Twenty-fifth International Conference on Machine Learning*, number July, pages 1096–1103.

Vinciarelli, A. (2002). A survey on off-line cursive word recognition. *Pattern recognition*, 35(7) :1433–1446.

Vinciarelli, A. and Bengio, S. (2002). Offline cursive word recognition using continuous density hidden Markov models trained with PCA or ICA features. *Proceedings of the 16th International Conference on Pattern Recognition*, 3 :81–84.

Vinciarelli, A. and Luettin, J. (2000). Off-line cursive script Recognition based on continuous density HMM. In *Proceedintgs of the 7th International Workshop on Frontiers in Handwriting Recognition*, pages 493–498.

Vinciarelli, A. and Luettin, J. (2001). A new normalization technique for cursive handwritten words. *Pattern Recognition Letters*, 22(9) :1043–1050.

Vishwanathan, S., Schraudolph, N., Schmidt, M., and Murphy, K. (2006). Accelerated training of conditional random fields with stochastic gradient methods. In *International Conference on Machine Learning*, volume 148, pages 969–976.

Wallach, H. and Wallach, H. (2002). *Efficient Training of Conditional Random Fields*. PhD thesis.

Weber, G. (2009). Top Languages. *Languages Today*.

Weidman, W., Manry, M., and Yau, H. (1989). A comparison of a nearest neighbor classifier and a neural network for numeric handprint character recognition. *International Joint Conference on Neural Networks*, pages 117–120.

Werbos, P. J. (1990). Backpropagation through time : What it does and how to do it. *Proceedings of the IEEE*, 78(10) :1550—-1560.

Williams, R. and Zipser, D. (1989a). Experimental analysis of the real-time recurrent learning algorithm. *Connection Science*, 1(1) :87–111.

Williams, R. and Zipser, D. (1990). Gradient-based learning algorithms for recurrent connectionist networks. In *Backpropagation : theory, architectures, and applications*. Citeseer.

Williams, R. J. and Peng, J. (1990). An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, pages 1–12.

Williams, R. J. and Zipser, D. (1989b). A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2) :270–280.

Xu, L., Krzyzak, A., and Suen, C. (1992). Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man and Cybernetics*, 22(3) :418–435.

Young, S., Evermann, G., Gales, M., Hain, T., Kershaw, D., Moore, G., Odell, J., Ollason, D., Povey, D., Valtchev, V., and Woodland, P. (2006). *The HTK book*.

Zhu, C. (1994). L-BFGS-B fortran subroutines for large-scale bound constrained optimization. *ACM Transactions on Mathematical Software*, pages 1–17.

Zissman, M. (1996). Comparison of four approaches to automatic language identification of telephone speech. *Transactions on Speech and Audio Processing*, 4(1).