Image and video text recognition using convolutional neural networks

Zohra Saidane

Thèse présentée et soutenue publiquement le 16 décembre 2008 pour l'obtention du grade de

DOCTEUR TELECOM PARISTECH

SPÉCIALITÉ : SIGNAL ET IMAGE

Reconnaissance de texte dans les image and les videos en utilisant des réseaux de neurones à convolutions

Autheur:
Zohra SAIDANE

JURY

Président	Alain MARUANI	Professeur, Télécom ParisTech
Rapporteur	Faouzi GHORBEL	Professeur, ENSI Tunis
Rapporteur	Olivier LEZORAY	MC HDR, GREYC UMR
Co-directeur	Christophe GARCIA	Expert émérite, Orange Labs
Co-directeur	Jean Luc DUGELAY	Professeur, EURECOM

December 2008

Acknowledgements

This thesis owes its existence to the help and support of many people.

First, I would like to express my deepest appreciation and gratitude to my supervisor Christophe Garcia for his advice and permanent guidance throughout this research. I enjoyed working with him and learning from him.

I am also indebted to my academic supervisor Jean Luc Dugelay. I was fortunate to have him as my Professor during my undergraduate studies and have his support during all these years.

I would like to thank the members of my jury; in particular Faouzi Ghorbel and Olivier Lezoray, who took the time and energy to point out my errors and omissions. My thanks also go to Alain Maruani for honoring me by presiding my jury.

To all members of the ICM group[1] at Orange Labs, I am very grateful for the cooperative spirit and the excellent working atmosphere, I really spent three very pleasant years.

Finally, I want to say thank you to my family for their continuing support in every respect and especially to my Ayoub, and my little Cheima.

[1] ICM group stands for Multimedia Content Indexing group

Résumé

Grâce à des moyens de stockage de plus en plus puissants, les ressources multimédia sont devenues de nos jours des ressources incontournables, aussi bien dans le domaine de l'information et de l'audiovisuel (agences de presse, INA), que de la culture (musées), des transports (surveillance), de l'environnement (images satellitaires), ou de l'imagerie médicale (dossiers médicaux en milieux hospitaliers). Ainsi, le défi est-il de comment trouver rapidement l'information pertinente. Par conséquent, la recherche en multimédia est de plus en plus concentrée sur l'indexation et la récupération de l'information. Pour accomplir cette tâche, le texte inclus dans les images et les vidéos peut être un élément clé pour l'indexation. Les défis de la reconnaissance du texte dans les images et les vidéos sont nombreux : mauvaise résolution, caractères de tailles différentes, artéfacts dus à la compression et aux effets d'anti-recouvrement, arrière plan complexe et variable.

Il y a quatre étapes pour la reconnaissance du texte: (1) détection de la présence du texte, (2) localisation de la région du texte, (3) extraction et amélioration du texte, et finalement (4) la reconnaissance du contenu du texte.

Dans ce travail nous nous concentrerons sur cette dernière étape et supposerons donc que la zone de texte à été détectée, localisée et extraite correctement. Ce module de reconnaissance peut être aussi divisé en quelques sous-modules tel que: un module de binarisation de texte, un module de segmentation de texte et un module de reconnaissance de caractères.

Nous nous sommes intéressés aux réseaux de neurones à convolutions. Ce sont des réseaux de neurones dont la topologie est similaire à celle du cortex visuel des mammifères. Les réseaux de neurones à convolutions ont été initialement utilisés pour la reconnaissance de chiffres manuscrits. Ils ont ensuite été appliqués avec succès à de nombreux problèmes de reconnaissance de forme.

Nous proposons dans cette thèse la conception d'une nouvelle méthode de binarisation d'image de texte, la conception d'une nouvelle méthode de segmentation d'images de texte, l'étude d'un réseau de neurones à convolutions pour la reconnaissance d'images de caractères, une discussion sur la pertinence de l'étape de binarisation pour la reconnaissance de texte dans les images basée sur des méthodes d'apprentissage automatique, et la conception d'une nouvelle méthode de reconnaissance de texte dans les images basée sur la théorie des graphes.

Abstract

Thanks to increasingly powerful storage media, multimedia resources have become nowadays essential resources, in the field of information and broadcasting (News Agency, INA), culture (museums), transport (monitoring), environment (satellite images), or medical imaging (medical records in hospitals). Thus, the challenge is how to quickly find relevant information. Therefore, research in multimedia is increasingly focused on indexing and retrieval techniques. To accomplish this task, the text within images and videos can be a relevant key. The challenges of recognizing text in images and videos are many: poor resolution, characters of different sizes, artifacts due to compression and effects of anti-recovery, very complex and variable background.

There are four steps for the recognition of the text: (1) detecting the presence of the text, (2) localizing of the text, (3) extracting and enhancing the text area, and finally (4) recognizing the content of the text.

In this work we will focus on this last step and we assume that the text box has been detected, located and retrieved correctly. This recognition module can also be divided into several sub-modules such as a binarization module, a text segmentation module, a character recognition module.

We focused on a particular machine learning algorithm called convolutional neural networks (CNNs). These are networks of neurons whose topology is similar to the mammalian visual cortex. CNNs were initially used for recognition of handwritten digits. They were then applied successfully on many problems of pattern recognition.

We propose in this thesis a new method of binarization of text images, a new method for segmentation of text images, the study of a convolutional neural network for character recognition in images, a discussion on the relevance of the binarization step in the recognition of text in images based on machine learning methods, and a new method of text recognition in images based on graph theory.

Sommaire

Chapter 0

French Summary

0.1 Introduction

Grâce à des moyens de stockage de plus en plus puissants, les ressources multimédia sont devenues de nos jours des ressources incontournables, aussi bien dans le domaine de l'information et de l'audiovisuel (agences de presse, INA), que de la culture (musées), des transports (surveillance), de l'environnement (images satellitaires), ou de l'imagerie médicale (dossiers médicaux en milieux hospitaliers). Ainsi, le défi est-il de comment trouver rapidement l'information pertinente. Par conséquent, la recherche en multimédia est de plus en plus concentrée sur l'indexation et la récupération de l'information. Pour accomplir cette tâche, le texte inclus dans les images et les vidéos peut être un élément clé pour l'indexation. Les défis de la reconnaissance du texte dans les images et les vidéos sont nombreux, nous en citons:

- La résolution : elle varie entre 160×100 pixels et 720×480 pixels. Pour le format CIF (384×288 pixels) la taille d'un caractère est plus petite que 10 pixels tandis que pour les documents numériques, elle est entre 50 et 70 pixels.
- Les caractères sont de différentes tailles
- Des artéfacts qui sont dus à la compression et aux effets d'anti-recouvrement
- Le fond peut être très complexe et variable.

Il y a quatre étapes pour la reconnaissance du texte: (1) la détection de la présence du texte, (2) la localisation de la région du texte, (3) l'extraction et amélioration du texte, et finalement (4) la reconnaissance du contenu du texte.

Dans ce travail nous nous concentrerons sur cette dernière étape et supposerons donc que la zone de texte a été détectée, localisée et extraite correctement. Ce module de reconnaissance peut être aussi divisé en quelques sous modules tel que: un module de binarisation de texte, un module de segmentation de texte, un module de reconnaissance de caractères.

Ces sous-modules peuvent interagir différemment selon les contraintes de l'application voulues et les méthodes utilisées.

Nous nous sommes intéressés particulièrement aux réseaux de neurones à convolutions. Ce sont des réseaux de neurones dont la topologie est similaire à celle du cortex visuel des mammifères. En fait, ils se basent principalement sur trois idées:

- des champs réceptifs locaux qui sont utilisés pour détecter les caractéristiques visuelles élémentaires dans l'image tels que les bords et les coins;

- des poids partagés entre tous les emplacements possibles du champ réceptif de manière à extraire la même caractéristique de l'image d'entrée et qui a l'avantage de réduire le coût de calcul;

- des opérations de sous-échantillonnage en vue de réduire le coût de calcul et de la sensibilité aux transformations affines tels que les translations et les rotations.

Les réseaux de neurones à convolutions ont été initialement utilisés pour la reconnaissance de chiffres manuscrits. Ils ont ensuite été appliqués avec succès à de nombreux problèmes de reconnaissance de forme. Deux exemples bien connus, où ces réseaux ont été appliqués, sont LeNet-5 et CFF.

LeNet-5 a été proposé par LeCun et al. [LBBH98] pour la reconnaissance de texte manuscrit et il est actuellement présent dans plusieurs banques pour effectuer la reconnaissance de chèques bancaires. CFF est un système de détection de visage proposé par Garcia et Delakis [GD04]. Il est très robuste aux différents types de bruit et de distorsions qu'on trouve généralement dans les images de visage. Les résultats de ce système sont supérieurs aux résultats des méthodes de l'état de l'art évaluées sur des bases de données publiques. Ces performances nous ont encouragé à étudier les réseaux de neurones dans le cadre du problème de reconnaissance de texte dans les images et les vidéos.

Nous proposons dans cette thèse:

- La conception d'une nouvelle méthode de binarisation d'images de texte basée sur une nouvelle architecture de réseau de neurones à convolutions qui traite les images couleur de texte et produit directement les images binaires correspondantes.

- La conception d'une nouvelle méthode de segmentation d'images de texte. Cette méthode est également basée sur une nouvelle architecture de réseau de neurones à convolutions qui traite des images couleur de texte. Pour chaque image traitée, cette méthode produit un vecteur qui indique les positions frontière entre les caractères consécutifs.
- L'étude d'un réseau de neurones à convolutions pour la reconnaissance d'images de caractères.
- Une discussion sur la pertinence de l'étape de binarisation pour la reconnaissance de texte dans les images basée sur des méthodes d'apprentissage automatique.
- La conception d'une nouvelle méthode de reconnaissance de texte dans les images basée sur la théorie des graphes. Cette méthode fait appel au système de segmentation et au système de reconnaissance de caractères exposés précédemment.

0.2 Système de binarisation

Les systèmes de reconnaissance optique de caractères actuels (ROCs) nécessitent une étape de binarisation qui vise à séparer les pixels de texte des pixels de l'arrière-plan de l'image traitée. En fait, la plupart des systèmes ROCs ne fonctionnent que sur des images binaires. La plus simple façon pour obtenir une image binaire est de choisir une valeur seuil, puis de classer tous les pixels dont les valeurs sont au-dessus de ce seuil comme étant de pixels d'arrière plan, et tous les autres pixels comme étant de pixels de texte. Le problème alors est de savoir comment sélectionner le bon seuil.

La plupart des méthodes de binarisation des images de texte se basent sur des méthodes de seuillages globales ou locales. Ces seuils sont déterminés en fonction de quelques statistiques concernant les distributions de la luminance ou de la chrominance généralement basées sur des histogrammes, sans tenir compte de la forme des caractères.

Nous proposons un système de binarisation d'images de texte couleurs qui permet:

- le développement d'un système de binarisation des images de textes par apprentissage supervisé sans aucun choix empirique de paramètre;
- la classification automatique de chaque pixel en pixel de texte ou pixel de fond, à partir d'extracteurs de caractéristiques locales et globales automatiquement appris, permettant de prendre en compte directement la forme des caractères;

- en plus de la forme, la prise en compte directe et simultanée des différents canaux de couleur, sans réduire l'analyse à la seule luminance ou à un traitement indépendant par canal;

- la robustesse aux bruits, aux faibles contrastes, aux variations de chrominance, aux effets de transparence et à la complexité du fond.

Notre approche repose sur une architecture neuronale composée de plusieurs couches hétérogènes. Ces couches permettent à la fois de développer automatiquement des extracteurs de caractéristiques bas niveau, caractérisant les formes possibles des caractères, et d'apprendre des règles permettant de les séparer du fond afin d'étiqueter automatiquement les pixels de l'image en pixel de texte ou pixel de fond. Tous les poids de connexions des neurones sont réglés au cours d'une phase d'apprentissage, à partir d'un ensemble d'images de textes, dont les images binaires correspondantes sont connues.

L'architecture neuronale agit par la suite comme une cascade de filtres permettant de transformer, une zone d'image couleur contenant le texte à binariser en une carte numérique, de la taille de la zone d'image en entrée, dont les éléments sont compris entre -1 et 1 et dans laquelle les pixels "texte" porte une valeur négative et les pixels "fond" une valeur positive.

Architecture du système de binarisation

Le réseau de neurones mis en place est constitué de cinq couches hétérogènes interconnectées comme le montre la figure 1. Les couches contiennent une série de cartes issues d'une succession d'opérations de convolution et de sous-échantillonnage pour le calcul des caractéristiques et de sur-échantillonnage et de convolution inverse pour la construction de l'image binaire. Elles réalisent, par leurs actions successives et combinées, l'extraction de primitives (forme et couleur) dans l'image en entrée pour conduire à la production d'une image binaire.

Plus précisément, l'architecture proposée contient:

- une couche d'entrée E: constituée de N_E cartes E_c correspondant aux N_E canaux de couleur de l'image d'entrée selon l'espace de codage des couleurs choisi (par exemple, les modes RGB ou YUV avec $N_E = 3$). Chaque carte représente ainsi une matrice image de taille $H \times L$ où H est le nombre de lignes et L le nombre de colonnes de l'image d'entrée. Pour chaque pixel $(P_{i,j})_c$ d'un canal de couleur c donné de l'image ($(P_{i,j})_c$ variant de 0 à 255), l'élément correspondant de la carte E_c, $(E_{i,j})_c = ((P_{i,j})_c - 128)/128$, est compris entre -1 et 1.

- une couche de convolution C_1, constituée de N_{C1} cartes $C_{1,i}$. Chaque carte $C_{1,i}$ est connectée à toutes les cartes d'entrée E_c. Les neurones

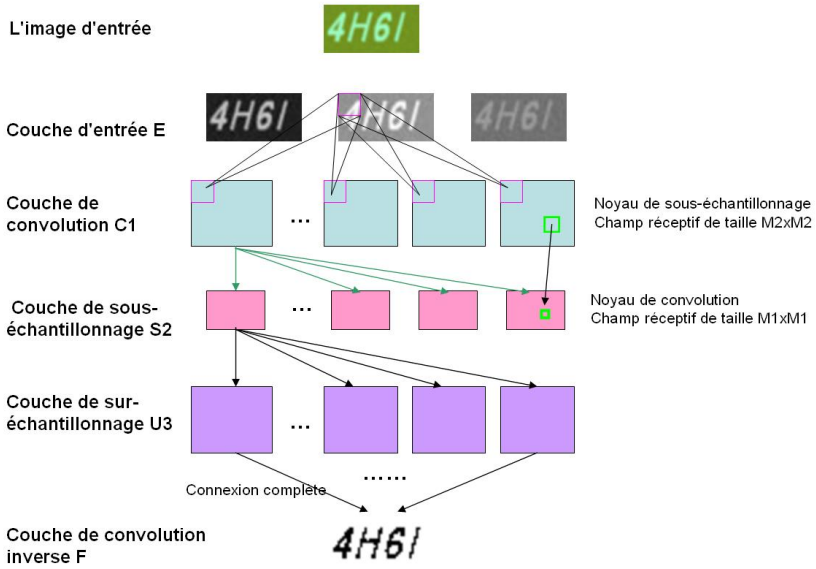


Figure 1: Architecture du système de binarisation

d'une carte $C_{1,i}$ sont linéaires. Chacun est connecté par des synapses à un ensemble de $M_1 \times M_1$ éléments voisins (champs réceptif) dans chaque carte E_c. Il reçoit un biais. Ces $M_1 \times M_1$ synapses plus le biais sont partagées par l'ensemble des neurones de $C_{1,i}$. Chaque carte $C_{1,i}$ correspond donc au résultat d'une convolution par un noyau $M_1 \times M_1$ augmentée d'un biais, dans les cartes d'entrée. Cette convolution se spécialise en un détecteur de certaines formes bas niveaux dans les cartes d'entrée comme des coins, des lignes de contraste orientées, etc. Chaque carte $C_{1,i}$ est donc de taille $H_1 \times L_1$ où $H_1 = (H - M_1 + 1)$ et $L_1 = (L - M_1 + 1)$, pour empêcher les effets de bord de la convolution.

- une couche de sous-échantillonnage S_2, constituée de N_{S2} cartes $S_{2,j}$. Chaque carte $S_{2,j}$ est connectée à toutes les cartes $C_{1,i}$. Chaque neurone d'une carte $S_{2,j}$ reçoit la somme des moyennes de $M_2 \times M_2$ éléments voisins (champs réceptif) dans les cartes $C_{1,i}$ multipliées par un poids synaptique, et ajoute à cette somme un biais. Poids synaptique et biais, à apprendre, sont partagés par l'ensemble des neurones de chaque carte. La sortie de chaque neurone est obtenue après passage dans une fonction sigmoïde. Chaque carte $S_{2,j}$ est de taille $H_2 \times L_2$ où $H_2 = H_1/M_2$ et $L_2 = L_1/M_2$.

  A ce niveau, des caractéristiques de l'image sont extraites et condensées.

Les phases suivantes vont permettre de construire l'image binaire de sortie.

- une couche de sur-échantillonnage U_3, constituée de N_{U3} cartes $U_{3,k}$. Chaque carte $U_{3,k}$ est connectée à toutes les cartes $S_{2,j}$ de la couche S_2. Chaque groupe de $M_2 \times M_2$ neurones de la carte $U_{3,k}$ est connecté par des synapses à un neurone dans chaque carte $S_{2,j}$. Chaque neurone de la carte $U_{3,K}$ multiplie la sortie des neurones de $S_{2,j}$ auxquels il est connecté par un poids synaptique et y ajoute un biais. La sortie de chaque neurone est obtenue après passage dans une fonction sigmoïde. Les $M_2 \times M_2$ poids synaptiques et biais, à apprendre, sont partagés par l'ensemble des neurones de chaque carte. Chaque carte $U_{3,k}$ est de taille $H_3 \times L_3$ où $H_3 = H_2 \times M_2 = H_1$ et $L_3 = L_2 \times M_2 = L_1$.

- une couche de convolution inverse F, constituée d'une seule carte de sortie. Cette carte est connectée à toutes les cartes de la couche de sur-échantillonnage U_3. Chaque neurone d'une carte $U_{3,k}$ est connecté par des synapses à un ensemble de $M_1 \times M_1$ éléments voisins dans la carte F. Il s'agit du schéma inverse de la convolution où chaque neurone de la carte de convolution est connecté à un ensemble $M_1 \times M_1$ éléments voisins dans les cartes d'entrées. Les $M_1 \times M_1$ synapses utilisées plus le biais sont partagées par l'ensemble des neurones de la carte finale. La sortie de chaque neurone est obtenue après passage dans une fonction sigmoïde. La convolution inverse permet de construire la carte finale à la taille de l'image d'entrée: elle est de taille $H_4 \times L_4$ où $H_4 = (H_3 + M_1 - 1) = (H_1 + M_1 - 1) = H$ et $L_4 = (L_1 + M_1 - 1) = L$. Cette carte représente donc la matrice des valeurs de l'image binaire reconstruite. Etant donné que les valeurs de sortie des neurones sont continues et se situent dans l'intervalle $[-1, 1]$, on évalue le signe de chaque élément pour convertir la matrice obtenue en image binaire. Soit $f_{i,j}$ la valeur d'un élément à la position i, j dans la carte F. Le pixel $P_{i,j}$ à la position i, j de l'image binaire est attribué soit la valeur zéro si $f_{i,j} < 0$ (texte, noir), soit la valeur 255 si $f_{i,j} \geq 0$ (fond, blanc).

Apprentissage du système de binarisation

L'étape suivante consiste à constituer une base d'apprentissage d'images annotées de façon à régler par apprentissage les poids des synapses de tous les neurones de l'architecture.

Pour ce faire, on procède comme suit:

- un ensemble T d'images de textes binaires de taille $H \times L$ est construit. A partir de chaque image binaire, on construit une image de couleur en

choisissant la couleur de texte et celle du fond, et on applique différents types de bruits (bruits uniformes, bruits gaussiens), ainsi que des filtres de lissage pour obtenir des images texte de synthèse se rapprochant le plus possible des images de texte réels et exhibant une forte variabilité. Pour chaque image ainsi construite, on dispose donc de l'image binaire correspondante qui constituera l'image de sortie désirée du réseau de neurones. L'ensemble T est appelé l'ensemble d'apprentissage.

- On apprend automatiquement l'ensemble des poids synaptiques et des biais de l'architecture neuronale. Pour cela, ces éléments sont tout d'abord initialisés de manière aléatoire (petites valeurs). Les N_T images I de l'ensemble T sont ensuite présentées de manière aléatoire à la couche d'entrée du réseau de neurone, sous forme de N_E cartes qui correspondent aux N_E canaux couleur de l'image selon l'espace de couleur choisi ($N_E = 3$ en général). Pour chaque image I en entrée, on présente les valeurs de l'image binaire correspondante normalisées entre -1 et 1 comme étant les valeurs désirées D_h vers lesquelles les neurones F_h de la dernière couche doivent converger. La couche d'entrée et les couches C_1, S_2, U_3, et F sont activées l'une après l'autre. En couche F, on obtient la réponse du réseau de neurones à l'image I. Le but est d'obtenir des valeurs F_h identiques aux valeurs désirées D_h. On définit une fonction objectif à minimiser pour atteindre ce but:

$$O = \frac{1}{N_T \times L \times H} \sum_{k=1}^{N_T} \sum_{h=1}^{L \times H} (F_h - D_h)^2 \tag{1}$$

Il s'agit donc de minimiser l'erreur moyenne quadratique entre les valeurs produites et désirées sur l'ensemble des images annotées de l'ensemble d'apprentissage T. Pour minimiser la fonction objectif O, on utilise l'algorithme itératif de rétro-propagation du gradient qui va permettre de déterminer l'ensemble des poids synaptiques et biais optimaux.

Après la phase d'entrainement, le système est capable de produire une image binaire correspondante à une image de texte couleur comme suit:

1. Redimensionner l'image de texte à la hauteur de la rétine, tout en préservant le rapport entre la largeur et l'hauteur de l'image (la taille de la rétine est égale à la taille des images d'entrainement);

2. Présenter l'image redimensionnée à l'entrée du réseau;

3. Récupérer la carte de sortie du réseau;

4. La dernière étape consiste à convertir les valeurs de cette carte en une image binaire en fonction du signe des neurones de sortie. Soit $f_(i,j)$ la valeur de l'élément (i,j) dans le carte de sortie F. Le pixel $P_(i,j)$ à la position (i,j) dans l'image binaire est attribué à 0 si $f_(i,j) < 0$ et à 1 si $f_(i,j) \geq 0$.

Nous devons mentionner ici que l'un des avantages de ce système est que la largeur de l'image à traiter ne doit pas obligatoirement être égale à la largeur de la rétine, grâce à la propriété des poids partagés. En fait, ces matrices de poids partagés peuvent être appliquées autant de fois nécessaire en fonction de la largeur de l'image à binariser.

Pour tester les performances de notre méthode, nous utilisons deux bases de données:

- La première est composée d'images de synthèse, qui nous permettent d'évaluer la robustesse de notre méthode en fonction de bruit et des variations de contraste.

- La deuxième contient des images réelles recueillies à partir de vidéo et de pages Web.

Nous comparons notre méthode aux méthodes de binarisation bien connues suivantes : la méthode d'Otsu [Ots79], la méthode de Niblack [Nib86], la méthode de Sauvola [SSHP97], et la méthode de Lienhart [LW02]. L'évaluation est basée sur deux critères:

- l'erreur quadratique moyenne obtenue entre les images binaires résultats et les images binaires souhaitées pour les tests sur les images synthétiques concernant l'évaluation de la robustesse aux variations de contraste et au bruit;

- le taux de reconnaissance d'un logiciel gratuit nommé OCR Tesseract [Hp], et celui d'une version d'essai d'un logiciel commercial nommé Abby FineReader 8.0 [Abb] pour le test sur les images réelles.

Les résultats obtenus montrent la stabilité de notre méthode par rapport au bruit et à la variation de contraste. Les taux de reconnaissance enregistrés lorsque nous utilisons notre méthode de binarisation avec un logiciel ROC commercial est de 86.23%.

0.3 Système de segmentation

Quand la reconnaissance est basée sur les caractères, une étape de prétraitement appelé segmentation est généralement nécessaire. Cette étape, qui

consiste à déterminer les frontières de chaque caractère, influe fortement sur la performance globale des systèmes de reconnaissance de texte.

Dans le cas des images binaires, de bonnes résolutions, une analyse du profil de la projection verticale est suffisante pour obtenir de bons résultats. L'analyse de la projection verticale consiste à compter le nombre de pixels noirs dans chaque colonne de l'image de façon à détecter la zone de blanc qui sépare les caractères. Dans le cas d'images couleur, de mauvaise résolution, contenant du bruit et ayant un arrière plan compliqué, ce schéma de traitement n'est pas suffisant.

Cette étape est difficile pour deux raisons:

- d'une part les images de texte ont généralement un arrière-plan complexe contenant du bruit et de variations de luminance;
- d'autre part, une partie d'un caractère pourrait être confondue avec un autre caractère. Par exemple, une partie de la lettre 'M' pourrait être confondue avec la lettre 'n', et une partie de la lettre 'W' pourrait être confondue avec la lettre 'V'. De la même façon, quelques caractères voisins pourraient être confondus avec un autre caractère, par exemple, 'ri', qui pourraient être confondu avec 'n'.

Nous proposons dans ce travail un système qui permet:

- la segmentation des caractères des images de textes par apprentissage supervisé sans aucun choix empirique de paramètre et en une passe seulement.
- un traitement rapide utilisant seulement trois couches neuronales pour traiter des images couleurs difficiles
- la prise en compte directe et simultanée des différents canaux de couleur, sans réduire l'analyse à seulement la luminance ou à un traitement indépendant par canal

Notre approche repose sur une architecture neuronale composée de plusieurs couches de convolutions. Ces couches permettent de développer automatiquement des extracteurs de caractéristiques spécifiques au traitement ciblé et qui favorise ainsi la détection automatique des frontières entre les différents caractères d'une image de texte couleur. Tous les poids de connexions des neurones sont réglés au cours d'une phase d'apprentissage, à partir d'un ensemble d'images de textes, dont la segmentation correspondante est connue.

L'architecture neuronale agit par la suite comme une cascade de filtres permettant de transformer une zone d'image couleur contenant le texte à

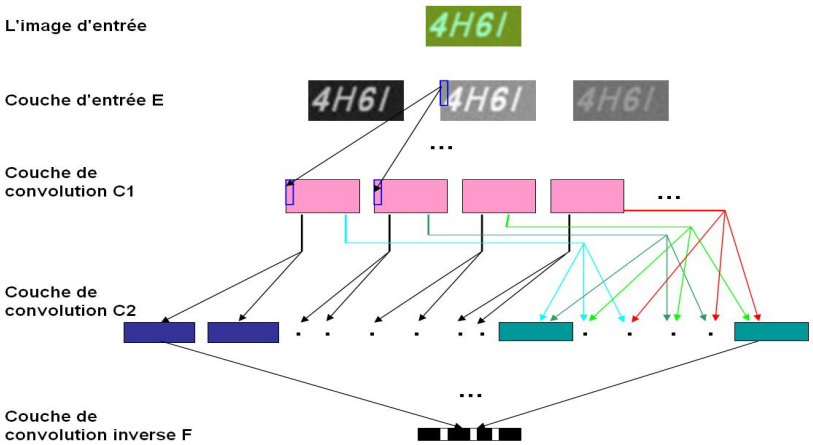


Figure 2: Architecture du système de segmentation

segmenter en une carte numérique, de la largeur de l'image en entrée moins quatre pixels dus aux effets de bords (deux de chaque côté), dont les éléments sont compris entre -1 et 1 et dans laquelle les pixels des frontières entre les caractères ont des valeurs positives et les autres ont des valeurs négatives.

L'architecture du réseau proposé se compose de trois couches en plus de la couche d'entrée. Chaque couche contient un ensemble de cartes caractéristiques qui sont le résultat de convolutions successives. L'application et la combinaison de ces opérations automatiquement appris assure l'extraction de caractéristiques robustes, ce qui conduit à la détection automatique des positions frontières entre les caractères consécutives dans une image de textes.

Architecture du système de segmentation

Le réseau de neurones mis en place est constitué de trois couches neuronales interconnectées comme le montre la figure 2. Les trois couches réalisent des opérations de convolutions permettant l'extraction de primitives (forme et couleur) dans l'image en entrée pour conduire à la production d'une carte (vecteur) encodant les frontières entre les différents caractères au niveau de la dernière couche du réseau.

Plus précisément, l'architecture proposée contient:

- une couche d'entrée E: constituée de N_E cartes E_c correspondant aux N_E canaux de couleur de l'image d'entrée selon l'espace de codage des

couleurs choisi (par exemple, les modes RGB ou YUV avec $N_E = 3$). Chaque carte représente ainsi une matrice image de taille $H \times L$ où H est le nombre de lignes et L le nombre de colonnes de l'image d'entrée. Pour chaque pixel $(P_{i,j})_c$ d'un canal de couleur c donné de l'image ($(P_{i,j})_c$ variant de 0 à 255), l'élément correspondant de la carte E_c, $(E_{i,j})_c = ((P_{i,j})_c - 128)/128$, est compris entre -1 et 1.

- une couche de convolution C_1, constituée de N_{C1} cartes $C_{1,i}$. Chaque carte $C_{1,i}$ est connectée à toutes les cartes d'entrée E_c. Chaque neurone d'une carte $C_{1,i}$ est connecté par des synapses à un ensemble de M_1 éléments voisins d'une même colonne dans chaque carte E_c. Il reçoit un biais. Ces M_1 synapses plus le biais sont partagées par l'ensemble des neurones de $C_{1,i}$. La sortie de chaque neurone est obtenue après passage dans une fonction sigmoïde. Chaque carte $C_{1,i}$ correspond donc au résultat d'une convolution par un masque vertical de M_1 éléments plus un biais, dans les cartes d'entrée. Chaque carte $C_{1,i}$ est donc de taille $H_1 \times L_1$ où $H_1 = (H - M_1 + 1)$ et $L_1 = L$, pour empêcher les effets de bord de la convolution.

- une deuxième couche de convolution C_2 constituée de N_{C2} cartes $C_{2,j}$. Ce nombre de cartes N_{C2} dépend de N_{C1} selon la formule suivante, $N_{C2} = 2 \times N_{C1} + N_{C1} \times (N_{C1} - 1)/2$. En effet, d'une part chaque carte $C_{1,i}$ est connectée à deux cartes $(C_{2,j}, C_{2,k})$ et d'autre part chaque paire de cartes $(C_{1,m}, C_{1,n})$ est connectée à une carte $C_{2,h}$. Chaque neurone d'une carte $C_{2,j}$ est connecté par des synapses à un ensemble de M_2 éléments voisins d'une même colonne dans chaque carte $C_{1,i}$ auquel il est connecté. Ces M_2 synapses plus le biais sont partagées par l'ensemble des neurones de $C_{2,j}$. La sortie de chaque neurone est obtenue après passage dans une fonction sigmoïde. Ce schémas de combinaison des connections entre les cartes assure l'extraction de primitives robuste aux bruits et à la mauvaise qualité des images à traiter. Chaque carte $C_{2,j}$ est donc de taille $H_2 \times L_2$ où $H_2 = (H_1 - M_2 + 1)$ et $L_1 = L$, pour empêcher les effets de bord de la convolution.

- une dernière couche de convolution F, constituée d'une seule carte de sortie. Cette carte est connectée à toutes les cartes de la couche précédente C_2. Chaque neurone de F est connecté par des synapses à un ensemble de $M_3 \times M_3$ éléments voisins dans chaque carte $C_{2,j}$. Les $M_3 \times M_3$ synapses utilisées plus le biais sont partagées par l'ensemble des neurones de la carte finale. La sortie de chaque neurone est obtenue après passage dans une fonction sigmoïde. Cette carte est donc de taille $H_3 \times L_3$ où $H_3 = (H_2 - M_3 + 1)$ et $L_3 = (L_2 - M_3 + 1)$, pour empêcher les

effets de bord de la convolution. La carte finale est en effet un vecteur ($H_3 = 1$) de largeur égale à celle de l'image en entrée moins quelques pixels ($M_3 - 1$ pixels) dues aux effets de bords. Ainsi, si un neurone de cette carte est activé (c'est-à-dire, sa sortie est positif), sa position est considérée comme étant une position frontière entre deux caractères successifs. Il suffit alors de découper l'image en entrée verticalement selon les positions des neurones activés pour obtenir les images correspondant aux caractères présents dans l'image en entrée.

Apprentissage du système de segmentation

L'apprentissage de ce système est fait en mode supervisé en utilisant une base d'apprentissage d'images annotées de façon à régler par apprentissage les poids des synapses de tous les neurones de l'architecture.

Pour ce faire, on procède comme suit:

- Un ensemble T d'images de textes couleurs de taille $H \times L$ sont construites en choisissant différentes couleurs et polices et en appliquant différents types de bruits (bruits uniformes, bruits gaussiens), ainsi que des filtres de lissage pour obtenir des images texte de synthèse se rapprochant le plus possible des images de texte réelles et exhibant une forte variabilité. Un fichier d'annotation est aussi construit en parallèle. Il indique pour chaque image les positions frontières entre les caractères qu'elle contient, ces positions correspondront aux neurones qu'on désire activer au niveau de la carte finale du réseau de neurones. L'ensemble T est appelé l'ensemble d'apprentissage.

- On apprend automatiquement l'ensemble des poids synaptiques et des biais de l'architecture neuronale. Pour cela, ces éléments sont tout d'abord initialisés de manière aléatoire (petites valeurs). Les N_T images I de l'ensemble T sont ensuite présentées de manière aléatoire à la couche d'entrée du réseau de neurones, sous forme de N_E cartes qui correspondent aux N_E canaux couleur de l'image selon l'espace de couleur choisi ($N_E = 3$ en général). Pour chaque image I en entrée, on présente un vecteur position contenant des 1 lorsque la position correspond à une frontière et -1 sinon comme étant les valeurs désirées D_h vers lesquelles les neurones F_h de la dernière couche doivent converger. La couche d'entrée et les couches C_1, C_2, et F sont activées l'une après l'autre. En couche F, on obtient la réponse du réseau de neurones à l'image I. Le but est d'obtenir des valeurs F_h identiques aux valeurs désirées D_h. On définit une fonction objectif à minimiser

pour atteindre ce but :

$$O = \frac{1}{N_T \times [L - (M_3 - 1)]} \sum_{k=1}^{N_T} \sum_{h=1}^{[L-(M_3-1)]} (F_h - D_h)^2 \qquad (2)$$

Il s'agit donc de minimiser l'erreur moyenne quadratique entre les valeurs produites et désirées sur l'ensemble des images annotées de l'ensemble d'apprentissage T. Pour minimiser la fonction objectif O, on utilise l'algorithme itératif de rétro-propagation du gradient qui va permettre de déterminer l'ensemble des poids synaptiques et biais optimaux.

Une fois l'entrainement terminé, le système est prêt à segmenter les images de texte couleurs comme suit:

1. On redimensionne l'image à segmenter à la taille $H \times L$ correspondant à la taille des cartes d'entrée du réseau lors de l'apprentissage.

2. On la place à l'entrée de l'architecture neuronale sous forme de k cartes qui correspondent aux k canaux couleur de l'image selon l'espace de couleur déjà choisi lors de l'apprentissage. Si l'image à traiter est une image en niveau de gris, il suffit de la répliquer k fois.

3. Les couche d'entrée et les couches C_1, C_2, et F sont activées l'une après l'autre.

4. En couche F, on obtient la réponse du réseau de neurones à l'image I, qui consiste en un vecteur de largeur égale à celle de l'image en entrée moins quelques pixels (à cause des effets de bords dû à la convolution). Les valeurs positives de ce vecteur indiquent les positions frontières entre les différents caractères de l'image.

5. Finalement, pour obtenir les images correspondantes aux différents caractères présents dans l'image texte initiale, il suffit de découper l'image de texte, verticalement, au niveau de ces positions avec ou sans une marge selon le système de reconnaissance qui va être utilisé.

Pour évaluer notre système de segmentation, nous l'avons entre autres testé sur une base de 52 images de texte collectées de pages web et de vidéo. Nous avons évalué manuellement le nombre de caractères correctement segmentés et nous avons enregistré un taux de segmentation correcte qui s'élève à 89.12%.

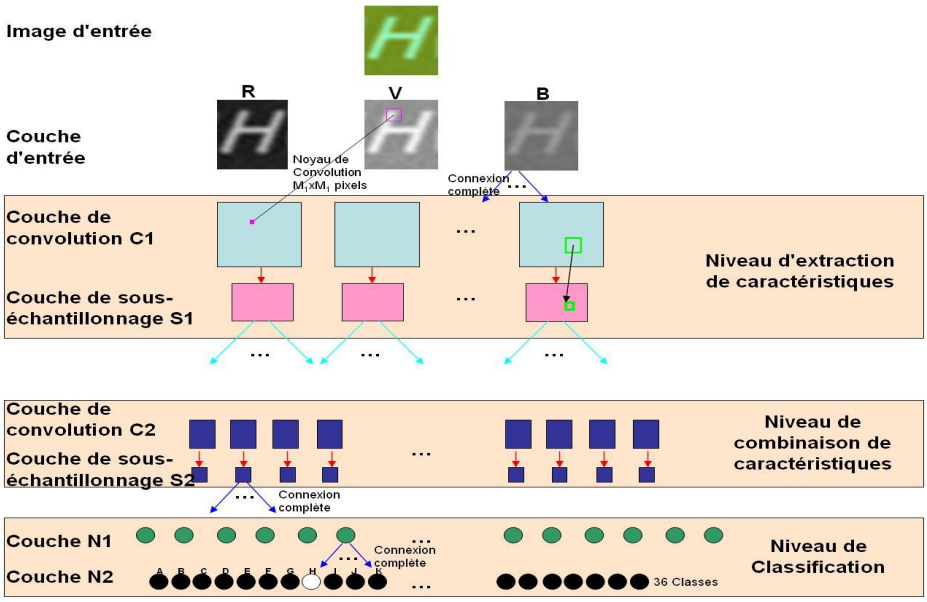


Figure 3: Architecture du système de reconnaissance de caractères

0.4 Système de reconnaissance de caractères

Dans la section précédente, nous avons présenté un système de segmentation qui pour une image de texte donné détecte les positions frontières entre les caractères consécutifs et ainsi permet l'extraction d'images de caractères prêtes à être reconnus.

Nous proposons donc un système de reconnaissance automatique de d'images couleurs de caractères, basé sur l'apprentissage supervisé, sans appliquer aucun prétraitement. En outre, notre système fait un usage direct des informations de couleur et assure la robustesse au bruit, aux arrières plans complexes et aux variations de luminance.

Architecture du système de reconnaissance de caractères

L'architecture du réseau de neurones est constituée de sept couches hétérogènes comme le montre la figure 3. Chaque couche contient plusieurs cartes caractéristiques qui sont le résultat de convolutions, sous-échantillonnages, ou des opérations unitaires de neurones. L'application et la combinaison de ces opérations, automatiquement apprises, assure l'extraction de caractéristiques robustes, ce qui conduit à la reconnaissance automatique de caractères dans les images difficiles.

Plus précisément, l'architecture proposée contient une couche d'entrée et

trois niveaux de traitement comme suit:

- Une couche d'entrée E: constituée de N_E cartes E_c correspondant aux N_E canaux de couleur de l'image d'entrée selon l'espace de codage des couleurs choisi (par exemple, les modes RGB ou YUV avec $N_E = 3$). Chaque carte représente ainsi une matrice image de taille $H \times L$ où H est le nombre de lignes et L le nombre de colonnes de l'image d'entrée. Pour chaque pixel $(P_{i,j})_c$ d'un canal de couleur c donné de l'image ($(P_{i,j})_c$ variant de 0 à 255), l'élément correspondant de la carte E_c, $(E_{i,j})_c = ((P_{i,j})_c - 128)/128$, est compris entre -1 et 1.

- Le niveau d'extraction de caractéristiques contenant les couches C_1 et S_2: La couche C_1 extraits des caractéristiques élémentaires directement à partir des trois canaux de couleur. En fait, cette couche est composée de N_{C1} de cartes. Il s'agit d'une couche de convolution où chaque unité de chaque carte est connectée à un $M_1 \times M_1$ unités dans chacun des canaux de couleur dans la couche d'entrée. En outre, les poids ajustables formant le champ réceptif, sont contraints à être égaux pour l'ensemble de la carte. Un biais ajustable est ajouté aux résultats de chaque masque de convolution. Ainsi, chaque carte peut être considérée comme une carte caractéristique. Plusieurs cartes conduire à l'extraction de multiples caractéristiques.

  Une fois qu'une caractéristique élémentaire est extraite, sa position exacte n'est plus importante, seulement sa position relative par rapport à d'autres caractéristiques est pertinente. Par conséquent, chaque carte de la deuxième couche cachée S_2 est le résultat d'une opération de sous-échantillonnage sur une carte correspondante dans la couche précédente C_1. Ainsi, la couche S_2 est composé de $N_{S2} = N_{C1}$ cartes. Nous utilisons cette sous-couche d'échantillonnage afin de réduire par deux la résolution spatiale qui réduit la sensibilité aux transformations, distorsions et variations d'échelle et de rotation.

- Le niveau de combinaison de caractéristiques contenant les couches C_3 et S_4: La couche C_3 permet d'extraire des informations plus complexes puisque ces cartes résultent de la combinaison de différents cartes caractéristiques de la couche S_2. Il s'agit de la deuxième couche de convolution dans ce système, chaque unité de chaque carte est connectée à un ensemble de $M_3 \times M_3$ unités dans les cartes de la couche S_2. Comme dans la couche C_1, les poids sont contraints à être égale pour l'ensemble de la carte. Un biais est ajouté aux résultats de chaque masque de convolution. Comme dans le premier niveau, la couche C_3

est suivie d'une couche sous-échantillonnage S_4, où chaque carte est le résultat d'opérations de sous-échantillonnage appliquées sur une carte correspondante dans la couche précédente C_3. En effet, cette réduction progressive de la résolution spatiale compensée par une augmentation progressive de la richesse de la représentation permet une invariance à un large degré de transformations géométriques.

- Le niveau de classification contenant les couches N_5 et N_6: Chacune de ces deux couches est entièrement connectée et contient des unités neuronales classiques. Le choix du nombre d'unités de N_5 est empirique, tandis que le nombre d'unités en N_6 dépend du nombre de classes à reconnaître. Si l'on considère les caractères alphanumériques, nous aurons 62 classes (26 caractères minuscules, 26 majuscules et 10 chiffres). Afin de réduire le nombre de classes, et par conséquent, le nombre de poids à apprendre, nous proposons d'utiliser seulement 36 classes, où majuscules et minuscules sont confondus. Ceci est possible grâce à la bonne capacité de généralisation de ce réseau.

Apprentissage du système de reconnaissance de caractères

Ce système est entraîné en mode supervisé en utilisant une base d'apprentissage d'images annotées de façon à régler par apprentissage les poids des synapses de tous les neurones de l'architecture.

Pour ce faire, on procède comme suit:

- Un ensemble d'apprentissage T d'images de textes couleurs de taille $H \times L$ sont construites en choisissant différentes couleurs et polices et en appliquant différents types de bruits (bruits uniformes, bruits gaussiens), ainsi que des filtres de lissage pour obtenir des images texte de synthèse se rapprochant le plus possible des images de texte réelles et exhibant une forte variabilité. Un fichier d'annotation est aussi construit en parallèle. Il indique pour chaque image la classe correspondante au caractère qu'elle contient.

- On apprend automatiquement l'ensemble des poids synaptiques et des biais de l'architecture neuronale. Pour cela, ces éléments sont tout d'abord initialisés de manière aléatoire (petites valeurs). Les N_T images I de l'ensemble T sont ensuite présentées de manière aléatoire à la couche d'entrée du réseau de neurone, sous forme de N_E cartes qui correspondent aux N_E canaux couleur de l'image selon l'espace de couleur choisi ($N_E = 3$ en général). Pour chaque image I en entrée, on présente un vecteur D_h vers lesquel le vecteur F_h de la dernière couche N_6 doit converger. Ce vecteur D_h contient les valeurs de sortie désirées qui sont

des -1 pour toute les neurones sauf pour le neurone correspondant à la classe de l'image d'entrée, qui est attribué 1. La couche d'entrée et les couches C_1, S_2, C_3, S_4, N_5 et N_6 sont activées l'une après l'autre. En couche N_6, on obtient la réponse du réseau de neurones à l'image I. Le but est d'obtenir des valeurs F_h identiques aux valeurs désirées D_h. On définit une fonction objectif à minimiser pour atteindre ce but :

$$O = \frac{1}{N_T \times 36} \sum_{k=1}^{N_T} \sum_{h=1}^{36} (F_h - D_h)^2 \tag{3}$$

Il s'agit donc de minimiser l'erreur moyenne quadratique entre les valeurs produites et désirées sur l'ensemble des images annotées de l'ensemble d'apprentissage T. Pour minimiser la fonction objectif O, on utilise l'algorithme itératif de rétro-propagation du gradient qui va permettre de déterminer l'ensemble des poids synaptiques et biais optimaux.

Une fois l'entrainement terminé, le système est prêt à reconnaître les images couleurs de caractères comme suit:

1. On redimensionne l'image à reconnaître à la taille $H \times L$ correspondant à la taille des cartes d'entrée du réseau lors de l'apprentissage.

2. On la place à l'entrée de l'architecture neuronale sous forme de k cartes qui correspondent aux k canaux couleur de l'image selon l'espace de couleur déjà choisi lors de l'apprentissage. Si l'image à traiter est une image en niveau de gris, il suffit de la répliquer k fois.

3. Les couche d'entrée et les couches C_1, S_2, C_3, S_4, N_5 et N_6 sont activées l'une après l'autre.

4. Finalement, le neurone de la couche N_6 qui a la valeur la plus importante donne directement la classe reconnue.

Pour évaluer la performance de la méthode proposée, nous utilisons la base publique ICDAR 2003 contenant des images de caractères compliquées. Notre système a atteint un taux de reconnaissance total de 84.53% qui dépasse les deux systèmes de l'état de l'art qui ont reporté des résultats sur cette base.

0.5 Discussion concernant l'importance de la binarisation

Nous avons présenté précédemment, deux systèmes différents pour la reconnaissance de texte dans les images. Le premier est basé sur une combinaison d'un système de binarisation et d'un système classique de reconnaissance optique de caractères, tandis que le second est basé sur une combinaison d'un système de segmentation d'images de texte et d'un système de reconnaissance de caractères.

À ce stade, une question qui se pose spontanément est *"est ce que l'étape de binarisation dans un système de reconnaissance de texte dans les images basé sur des algorithmes d'apprentissage est indispensable?"*.

En fait, d'une part, beaucoup de travaux dans l'état de l'art tentent de tirer profit des systèmes ROC classiques conçus à la base pour les documents imprimés. Ainsi, ils supposent que la binarisation est une étape nécessaire pour la reconnaissance de texte dans les images.

D'autre part, au cours des deux dernières décennies, la communauté de reconnaissance de texte dans les images a montré un intérêt croissant pour le traitement direct d'images en niveau de gris ou en couleur.

Malheureusement, il n'y a pas à ce jour des études sur la pertinence de la binarisation pour ce domaine précis. Nous souhaitons contribuer à l'étude de ce problème à travers la réalisation de quelques expériences qui permettent de déterminer quelle est la pertinence de l'étape de binarisation dans notre cas.

Dans nos expériences, nous avons choisi d'utiliser des images couleurs plutôt que des images en niveau de gris, parce que la couleur peut être un élément clé discriminant permettant de distinguer le texte sur les fonds complexes, ceci est particulièrement vrai lorsque les valeurs de luminance de certains pixels sont égaux alors que leurs valeurs de chrominance sont différentes.

Les différentes expérimentations effectuées nous ont mené aux conclusions suivantes:

- L'étape de binarisation n'est pas nécessaire dans le cas de reconnaissance basée sur les méthodes d'apprentissage. Le système de reconnaissance basé sur les réseaux de neurones à convolutions donne de bons résultats lors du traitement de l'image couleur directement.

- Le système de reconnaissance qui a appris sur des images couleurs effectue une meilleure généralisation des données synthétiques vers des

données réelles. C'est très intéressant car nous pouvons éviter le dur et long travail d'annotation.

- La capacité de généralisation est plus difficile en utilisant des images synthétiques binaires. En fait, le système se concentre principalement sur la forme des caractères, et étant donné qu'il ya un grand nombre de polices différentes dans les données réelles, il est tout à fait impossible de les représenter dans une base de données d'apprentissage.

Cela étant, la question de l'importance de l'étape de binarisation reste une question ouverte pour les autres méthodes de reconnaissance de texte dans les images et même dans le cas d'autres applications de reconnaissance de formes.

0.6 Graphe de reconnaissance de textes dans les images (iTRG)

Nous avons présenté jusqu'ici un système de segmentation des images de texte en caractères, ainsi qu'un système de reconnaissance des images de caractères. Il reste donc à concevoir un système global qui permet la reconnaissance de texte dans les images difficiles à cause d'une mauvaise résolution, d'un arrière plan complexe ou encore du bruit. Ce système se base d'une part sur les approches présentées précédemment et d'autre part sur la théorie des graphes.

Nous proposons un nouveau système de reconnaissance de textes dans les images à travers ce que nous avons appelé le "iTRG - image Text Recognition Graph". Il s'agit d'un graphe pondéré acyclique dirigé $G = (V, E, W)$, qui se compose d'un ensemble de sommets (ou nœuds) V, un ensemble d'arêtes (ou d'arcs) E et un ensemble de poids des arcs W. Dans le iTRG, les arcs sont orientés ce qui signifie qu'un arc $e(i, j)$ est orienté du nœud i au nœud j et que l'arc $e(j, i)$ n'existe pas. Les arcs du iTRG sont également pondérés, ce qui signifie qu'un poids est attribué à chaque arc. Dans notre cas, ce poids représente la probabilité de l'existence de l'arc correspondant, comme nous le verrons plus tard. Enfin, le iTRG est un graphe acyclique, ce qui signifie qu'il n'y a pas de cycles ou de boucles. Le système proposé consiste à construire le iTRG, puis à rechercher le meilleur chemin qui donne la plus probable séquence de caractères figurant dans l'image traitée.

Architecture du iTRG

Le système est basé sur cinq modules:

- Un module de sur-segmentation basé sur le système de segmentation présenté dans le rapport précédent, mais avec un post traitement particulier.
- Un module de construction des connections du graphe basé sur le résultat du module de sur segmentation.
- Un module de reconnaissance de caractères basé sur le système de reconnaissance présenté dans le rapport précédent mais qui dépend des connections du graphe.
- Un module de calcul des poids du graphe qui dépend du résultat de la sur segmentation et de la reconnaissance.
- Un module de recherche du plus court chemin dans ce graphe.

Ces modules sont présentés plus en détail ci-après:

1. Module de sur-segmentation : Ce module est basé sur le système de segmentation présenté précédemment qui prend en entrée une image de texte et produit un vecteur représentant la probabilité pour chaque colonne d'être une frontière entre deux caractères consécutifs. Mais, au lieu de considérer seulement les positions avec une probabilité au-dessus d'un seuil prédéfini, nous choisissons ici de définir un seuil plus bas et de considérer les positions dont les probabilités sont des maxima locaux.

   Ainsi, la sortie de ce module est un ensemble de positions frontière probables. Les plus pertinentes seront déterminées par le processus global détaillé dans les prochains paragraphes.

2. Module de construction de connections : Ce module prend le résultat du module de sur segmentation comme entrée. Ainsi, chaque position frontière possible est considérée comme un sommet dans le iTRG. Le but de ce module est de construire les connections entre les sommets. Au lieu de relier tous les sommets les uns aux autres, nous avons choisi de mettre certaines contraintes afin d'optimiser le calcul sans perdre d'efficacité. Ces contraintes sont les suivantes:

   - Un sommet i est connecté à un sommet j si la distance entre les deux est dans un intervalle prédéfini qui dépend bien sûr de la largeur des caractères.
   - Un sommet i ne sera pas connecté à un sommet j s'il existe un sommet k entre les deux dont la probabilité d'être une position frontière est au dessus d'un certain seuil.

3. Module de reconnaissance : Ce module est basé sur le système de reconnaissance de caractères présenté précédemment. Ce module prend en entrée les positions des sommets et les connexions du graphe. Chaque paire de sommets connectés correspond à une possible image de caractère. Ces images sont tronquées de l'original et à présenter à l'entrée du système de reconnaissance de caractères. Les résultats de ce module contribueront au calcul des poids du graphe.

4. Module de calcul des poids: Ce module prend en entrée les résultats du module de sur segmentation et du module de reconnaissance pour terminer la construction du iTRG. L'équation régissant les poids des arcs est détaillée ci-dessous:

$$\begin{aligned} edgeweight_{i,j} = \\ & outRec_{i,j} \times \\ & [outSegm(nod_i) \times outSegm(nod_j)] \times \\ & [dist_{(i,j)}] \end{aligned}$$

   Où:

   - $outRec_{i,j}$ représente la sortie du système de reconnaissance.
   - $outSegm(nod_i)$ représente la probabilité que le sommet i corresponde à une position frontière.
   - $dist_{(i,j)}$ représente la distance entre le sommet i et le sommet j en terme de nombre de sommets entre les deux.

5. Module de recherche du meilleur chemin : Dans la théorie des graphes, rechercher le meilleur chemin est souvent lié au problème de recherche du plus court chemin. Ce problème consiste à trouver un chemin entre deux sommets de telle sorte que la somme des poids de ses arcs est réduite au minimum. Nous avons choisi d'utiliser l'algorithme de Dijkstra qui est un algorithme de recherche du plus court chemin dans les graphes avec des poids positifs et qui est largement utilisé en raison de sa performance et son faible coût de calcul.

Dans notre cas, c'est la dernière étape du système de reconnaissance de texte. Une fois que nous avons traité l'image d'entrée avec le système de segmentation, construit les connections du graphe, traité chaque segment de l'image avec le système de reconnaissance de caractères, et calculé les poids des arcs, nous appliquons l'algorithme de Dijkstra pour récupérer la meilleure

séquence d'arcs. Les étiquettes de cette séquence donnent directement le texte reconnu. La figure 4 illustre avec un exemple complet les différentes étapes de reconnaissance de texte dans les images avec le iTRG.

Testé sur la base de donnée publique "ICDAR 2003 Robust Reading", le iTRG permet une amélioration de 10% sur toute la base et de 15% pour les images avec le moins de distorsions par rapport au schéma de combinaison simple du système de segmentation de texte avec le système de reconnaissance de caractères.

0.7 Conclusions et perspectives

Nous avons élaboré un système de reconnaissance robuste pour le texte contenu dans les images difficiles. Ce système est basé sur des modules qui s'appuient sur les réseaux de neurones à convolutions d'une part et sur la théorie des graphes d'autre part. Nous pensons qu'il est nécessaire de compléter ce système dans l'avenir avec un module de modélisation statistique de langage, en vue d'un système plus complet pour la reconnaissance des mots.

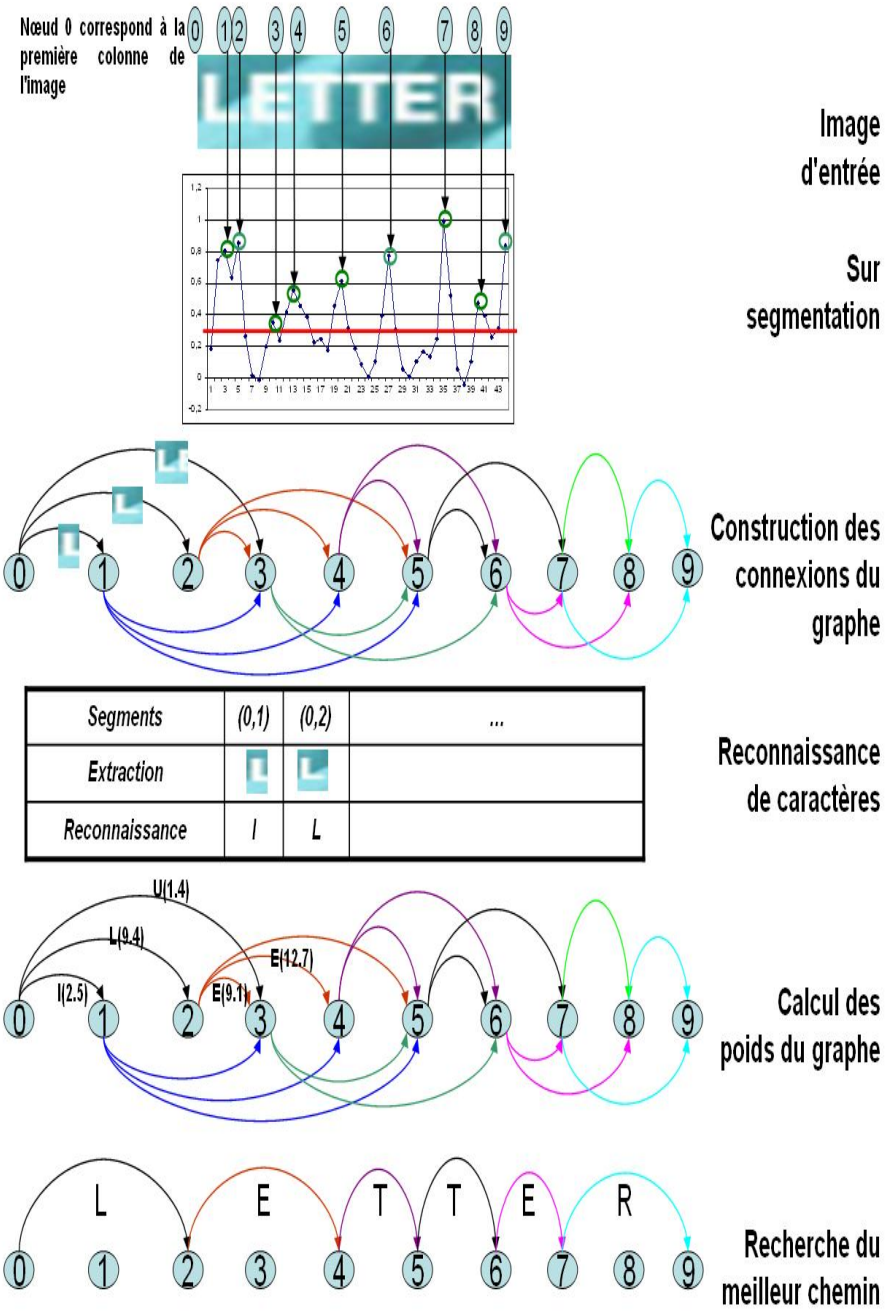


Figure 4: Architecture du iTRG

Chapter 1

Introduction

Celui qui trouve sans chercher est celui
qui a longtemps cherché sans trouver.

The one who finds without searching
is the one who searched for a long time without finding.
`[Gaston Bachelard] (1884 - 1964)`

Text recognition in images and video sequences is a field of research in pattern recognition, artificial intelligence and machine vision, usually called "Video OCR", which stands for Video Optical Character Recognition. Video OCR attempts to create a computer system which automatically detects, extracts and understands what is meant by the text embedded in the images and video frames. There are four steps to perform video text recognition (See figure 1.1):

1. Detection: detect the presence of text, which should answer to "Is there a text string in the current frame?"

2. Localization: localize the region of text, which should answer to "Where is the text string in the current frame?"

3. Extraction: extract the text, generally this step is accompanied with an enhancement processing.

4. Recognition: recognize the text, which should answer to "What does this text string say?", generally this step is accompanied with some binarization and/or segmentation pre-processing step.

Suppose you wanted to find nice web images containing the word 'Merci'. If you try this search on "Google image" for example, only 8 over the first

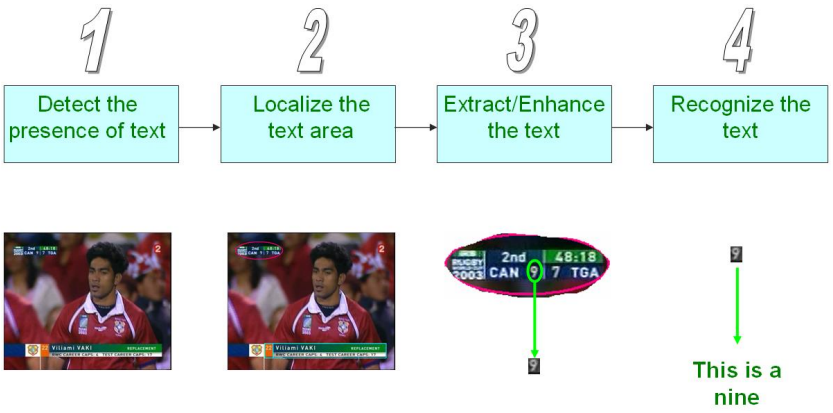


Figure 1.1: Video Text Recognition Scheme

18 images retrieved contain actually this word. This would not be the case, if the search is done inside the images instead of searching the text around them. To stress the motivation on this research work, we present some other interesting applications for text recognition in the following section.

1.1 Applications of video text recognition

First of all, we have to mention that machine vision and imaging is a booming market. A new machine vision market study, "Machine Vision Markets–2005 Results and Forecasts to 2010", published by the Automated Imaging Association (AIA), reports that machine vision and automated imaging continues to be a growth industry in North America. Moreover, Siemens "Pictures of the Future Fall 2006" stresses that, according to estimates provided by a number of manufacturers, the worldwide market volume for machine vision systems currently amounts to about 6,5-billion euro, with annual growth rates extending into the double-digit range. OCR systems represent 7% of the total sales (See figure 1.2 - VDMA 2006 (VDMA stands for Verband Deutscher Maschinen- und Anlagenbau - German Engineering Federation)).

Let us now, focus on some significant applications of Video OCR.

1.1.1 Multimedia Annotation and Retrieval

The tremendous increase in the use of multimedia not only as a communication medium, but also as an educational and entertaining medium, entails a need to powerful indexing system, so as to ensure easy access to the relevant information, navigation, and organization of vast repositories of multimedia

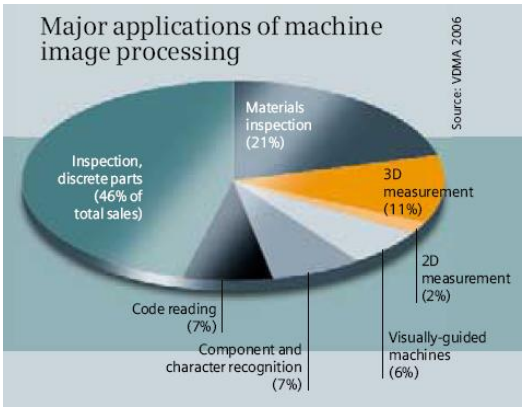


Figure 1.2: Machine Vision Applications

(audio, images, video). Manual annotation of video is extremely time consuming, and expensive in the face of ever growing video databases. Therefore, automatic extraction of video descriptions is desirable in order to annotate and search large video databases. Multimedia entities carry two type of information, content based information and concept based information. The first kind consists of low level features (colors, shapes, edges, frequencies, energy of the signal...), and could be automatically identified and extracted, where as the second consists of high level features, called also semantic features, which indicate what is represented in the image, or what is meant by the audio signal, and are in general hardly identified automatically. One of the goals of the multimedia indexing community is to design a system able of producing a rich semantic description of any multimedia document. Among the semantic features, the text embedded in images is of particular interest as it is abundant in videos with program credits and title sequences. In videos of news, text is often used as captions, and in sports videos game and player statistics are often superimposed on the frames in textual form. Video commercials ensure that the product and other shopping information are presented as readable text. When video text is automatically recognized, it not only provides keywords for annotation and search of image and video libraries but also aids in highlighting events which can then be used for video categorization, cataloging of commercials, logging of key events, and efficient video digest construction. Furthermore, video OCR can be used with other features, such as speech recognition and face identification or recognition to build a robust multi-modal system.

1.1.2 Library digitizing

The advantages of digital libraries are now widely recognized by commercial interests and public bodies alike. For example,

- the user of a digital library needs not to go to the library physically
- people can gain access to the information at any time, night or day
- the same resources can be used at the same time by a number of users
- an easy and rapid access to books, archives and images of various types

Therefore there are many collaborative digitization projects where advanced image and video OCR is used to digitize cover of journals and magazines.

1.1.3 Mobile phone services

The mobile phone is already becoming the most ubiquitous digital device across the world. Today, video and TV services are driving forward third generation (3G) deployment. Thus, many mobile phone services are being considered with interest, such as:

- Extracting and translating visual signs or foreign languages for tourist usage; for example, a handhold translator that recognizes and translates Asia signs into English or French.
- Managing personal and downloaded images and videos.

1.1.4 Robotics

Video OCR has also applications in robotics. In fact, with all of the textual indications, messages and signs we find in urban settings to provide us with all types of information, it is obviously interesting to give to robots reading capabilities of interpreting such information. For example, this can be used in driverless cars which may result in less-stressed 'drivers', higher efficiency, increased safety and less pollution (e.g. via completely automated fuel control).

1.2 Challenges

While the applications of video OCR are very interesting, this research faces many challenges. We list bellow some among the most prominent ones.

- Low resolution: it depends on the application and ranges from 160x100 pixels to 720x480 pixels. For CIF format (384x288 pixels) the size of a character is smaller than 10 pixels whereas for numerical documents, it is between 50 and 70 pixels, thus applying OCR techniques for video text recognition needs appropriate preprocessing.
- Characters are of different size, thus a multi resolution framework is necessary.
- Artifacts of anti aliasing. The videos are mostly produced in higher resolutions, then down sampled to be stored in database or transferred over networks, which creates visible aliasing artifacts. To avoid this, a low pass filter is applied, which disturbs the segmentation of the characters.
- Artifacts of compression. In fact, the MPEG compression scheme loses information which is estimated to be redundant for the human vision system. This scheme harms the recognition process, for example by disturbing the color uniformity.
- Background can be very complex and changeable, which makes the segmentation into foreground and background difficult.

1.3 History of Optical Character Recognition (OCR)

OCR involves reading text from paper and translating it into a form that the computer can manipulate such as ASCII codes (cf. figure 1.3). It can be seen as a process of three main steps:

1. Identification of text and image blocks: In general, we use white space to try to recognize the text in appropriate order. However, complex formatting such as cross-column headings or tables must be manually processed by 'zoning' (identifying and numbering text blocks) prior to OCR.

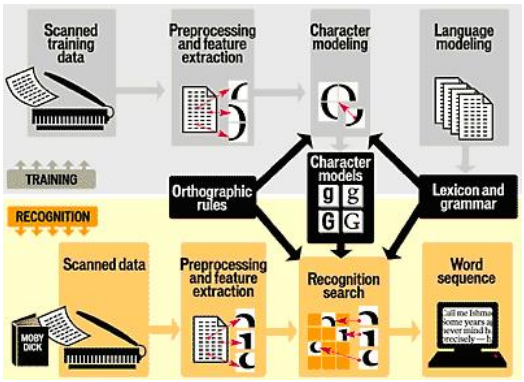


Figure 1.3: Different steps of OCR

2. Character recognition: The most common method used here is the feature extraction method, which identifies a character by analyzing its shape and building a model of features that distinguishes each character.

3. Word identification/recognition: Character strings are compared with appropriate dictionaries according to the language of the original document.

The first commercial machines were designed in the late 60's. Simple two-dimensional correlation techniques known as "Matrix Matching" were used. These techniques are based on the direct comparison of two bit map images. The unknown image needs to be compared with a set of templates representing known characters. Obviously, this method is not adapted to size and font variations, therefore topological recognition techniques based on a general description of the character's structure have been investigated. There are traditionally two methods for OCR: font-oriented and omni-font. Font-oriented algorithms have prior information about the characters to be recognized. Omni-font algorithms measure and analyze different characteristics of letters and classify them without any prior knowledge about their font and size. Omni-font methods can recognize bad quality characters because of their generalization abilities, but their recognition rate is usually lower than with font-oriented algorithms. Therefore, most systems employing omni-font recognition approaches do some form of automatic learning to improve their recognition ability; neural networks are widely used for this. Apart from that, it is CPU consuming and requires a large amount of memory. Nowadays, commercial OCR packages achieve a recognition rate ranging from 95% to 99% on common paper-printed documents. Intelligent character recognition (ICR) systems for recognizing constrained handwritten characters are

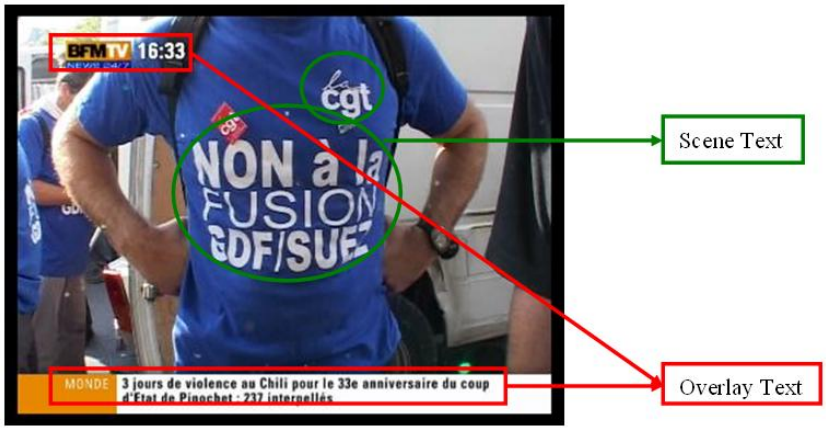


Figure 1.4: Overlayed Text vs. Scene Text

also commercially available. However, these systems require characters to be written in a similar way of printed characters. The recognition of unconstrained handwritten characters is still an area of research The performance of OCR systems closely relies on the quality of the targeted documents. This represents the main problem when applying OCR to video character recognition.

1.4 Properties of Video Text

It is important to analyse the characteristics and properties of text embedded in images and videos, keeping in mind that a lot of methods take advantages of them.

There are two kinds of text in videos (cf. figure 1.4): scene text, which was recorded with scene, such as, street name, car plate, the number on the back of the sports player etc., and artificial text which was added to the scene and often carries important information.

Both of them can present several variations with respect to the following properties:

Size In the same image or frame, we can find text of different size. This makes hard the choice of the system parameters (window size of some filters, size of morphological operators, some thresholds...). To cope with this problem, some works apply a fixed scale scheme to rescaled input images of different resolution [LDK00, LW02, WMR99], so that when the size of the text in the rescaled image fit that of the scheme,

it will be correctly processed. However still remains the problem of determining the size of the rescale.

Edge Most caption and scene texts are designed to be easily read, thereby resulting in strong edges at the boundaries of text and background. This property is used to detect or to localize text either directly by applying edge detection filters [KHE05a, LW02, CCC+04, SKHS98, AD99, GAK98] or indirectly by computing wavelet [LDK00] or Gabor coefficient [CYZW04, CSB01, TA03] or the Laplacian of Gaussian [CYZW04]. However, in the case of complex background, these methods can generate a lot of false alarms; therefore, non-text components have to be filtered out using size, thickness, aspect ratio, and gray-level homogeneity or using a classification system to classify the regions on text regions or non text regions.

Color Most of the times, the characters tend to have the same or similar colors. Therefore, some research has concentrated on finding text strings of uniform colors [Lie96, LE98, ZKJ95] using connected component based approach for text detection. However, video images and other complex color documents can contain text strings with various colors, i.e. different colors within one word, which requires a dedicated processing.

Aspect Ratio Characters do not have the same aspect ratio even when we consider the same font, which increases the difficulty of some tasks such as segmentation. Nevertheless, the methods based on the aspect ratio to filter out non text [SKHS98, GAK98] are not concerned by this fact, since they predefine a certain range for this value.

Inter-character distance Characters in a text line have a uniform distance between them. Thus, some methods use this property to segment a text line into words, such as the methods based on projection profiles [KHE05a, LW02, SKHS98], which are fast and give good results when the background is not too complex, otherwise they can be combined to other approaches.

Alignment Generally, the text lies horizontally. This alignment can be a good feature for scene text. Indeed, it can provide some cues to recover deformation of scene text, which has various perspective distortions [ZKJ95].

Motion Depending on the standard used, a video contains either 25 or 30 frames per second, so that every character should exist in many consec-

utive frames. The caption text can be static or can have a movement. In the latter case, it usually moves in a uniform way: horizontally or vertically, while scene text can have arbitrary motion due to camera or object movement. For caption text, the motion property can be used for tracking and enhancement [LDK00, LW02, KCC00].

1.5 Properties of other different types of text documents

While dealing with character recognition, we have to mention that there are different types of text documents, each of which has some particular characteristics compared to the others and some common characteristics with the others, therefore, some approaches can be applied to a group of them, and some others are specific to a given kind of text document.

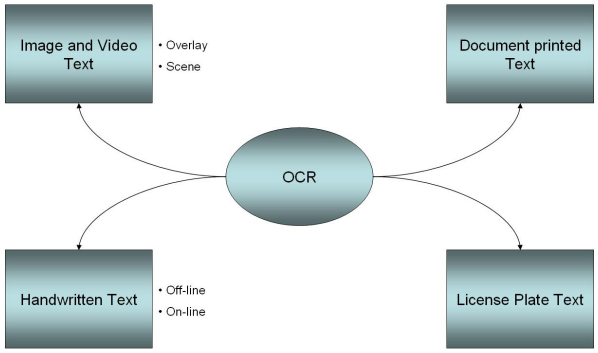


Figure 1.5: Different OCR Text Types

1.5.1 Printed text

Data in a paper document are usually captured by optical scanning and stored in a file of picture elements, called pixels, which are sampled in a grid pattern throughout the document. During the scanning stage a resolution of 300 - 400 dpi is used, which results in large image sizes (100 MB for a grayscale A4 page scanned at 400 dpi). The files are stored uncompressed or

compressed without loss, which keeps the full image quality and prevents artifacts. Moreover, document images contain mostly monochromatic text, and the background is not complex resulting in a segmentation process which can be performed in many cases based on a simple thresholding on the grayscale image. Documents which contain components that cannot be processed by OCR, such as graphics and halftones are preprocessed so that such fields are stored as separate elements, usually in some compressed format. Here are some properties of machine printed Latin text [Nag92]:

- Printed lines are parallel and roughly horizontal.
- The baselines of characters are aligned.
- Each line of text is set in a single point size.
- Ascenders, descenders, and capitals have consistent heights.
- Typefaces (including variants such as italic or bold) do not change within a word.
- Within a line of text, word spaces are larger than character spaces.
- The baselines of text in a paragraph are spaced uniformly.
- Each paragraph is left-justified or right-justified (or both), with special provisions for the first and last line of a paragraph.
- Paragraphs are separated by wider spaces than lines within a paragraph, or by indentation.
- Illustrations are confined to rectangular frames.
- In multicolumn formats, the columns are of the same width.

Commercial OCR packages achieve a recognition rate ranging from 95% to 99% on common paper-printed documents.

1.5.2 Handwritten text

An important property of handwritten text is that each person has its own handwriting style. Despite these variations each character has a basic specific shape which enables distinguishing it from the others. While humans can recognize easily handwritten characters, there is no OCR with the same level of performance. Some software such as “SimpleOCR” proposes to learn the handwriting style of the user. It is recommended to supply between 300 and

500 words (about 3-5 pages) of handwritten text. There are two schemes to perform handwritten recognition, the online scheme and the offline scheme.

1. On-line handwriting recognition means that the machine recognizes the writing while the user writes. An advantage of on-line devices is that they capture the temporal or dynamic information of the writing. This information consists of the number of strokes, the order of the strokes, the direction of the writing for each stroke, and the speed of the writing within each stroke. Most on-line transducers capture the trace of the handwriting or line drawing as a sequence of coordinate points. Online systems for handwriting recognition are available in hand held computers such as PDAs.

2. Offline handwriting recognition is performed after the completion of the writing. Words can be formed by discrete characters, continuous and discrete cursive forms (group of characters written with a single continuous motion), or a combination of them. Therefore, it usually requires costly and imperfect preprocessing to extract contours and to thin or skeletonize them. However, although off-line systems are less accurate than on-line systems, they are now good enough to have a significant economic impact on specialized domains such as interpreting handwritten postal addresses on envelopes and reading courtesy amounts on bank checks. In fact, the limited vocabulary in these latter domains helped a lot to improve the performance of such systems.

1.5.3 License plate characters

License plate recognition (LPR) is a particular kind of image text recognition system, since an image of the license plate is captured by a camera and sent to the recognizer. However, license plates have two main properties which should help to improve the performance of the recognition system. First, the possible colors of a license plate are limited. Second, the number of characters per license plate is fixed. Of course, since the vehicle plates are based on different country standards, i.e. that the two latter properties are different from one country to another, the LPR systems are country specific and are adapted to the country where they are installed and used. Still remain however, classical image processing problems such as:

- The poor image resolution, either because the plate is too far away or sometimes resulting from the use of a low-quality camera.

- Blurry images, particularly motion blur and most likely on mobile units

- Poor lighting and low contrast due to overexposure, reflection or shadows
- An object obscuring (part of) the plate, quite often a tow bar, or dirt on the plate
- A different font, popular for vanity plates (some countries do not allow such plates, eliminating the problem).

Nowadays there are a many commercial LPR systems suppliers, such as: Adaptive Recognition Hungary (Budapest Hungary), Asia Vision Technology Ltd. (Kowlong Tong Hong Kong), AutoVu Technologies, Inc. (Montréal, Québec Canada), Zamir Recognition Systems Ltd. (Jerusalem Israel)...
There is a lot of application for such systems, here are some of them:

- Parking: the plate number is used to automatically enter pre-paid members and calculate parking fee for non-members (by comparing the exit and entry times).
- Access Control: a gate automatically opens for authorized members in a secured area, thus replacing or assisting the security guard. The events are logged on a database and can be used to search the history of events.
- Tolling: the car number is used to calculate the travel fee in a toll-road, or used to double-check the ticket.
- Border Control: the car number is registered in the entry or exits to the country, and used to monitor the border crossings. It can short the border crossing turnaround time and cut short the typical long lines. Each vehicle is registered into a central database and linked to additional information such as the passport data. This is used to track all border crossings.
- Stolen cars: the system is deployed on the roadside, and performs a real-time match between the passing cars and the list of stolen cars. When a match is found a siren or display is activated and the police officer is notified with the detected car and the reasons for stopping the car.
- Enforcement: the plate number is used to produce a violation fine on speed or red-light systems. The manual process of preparing a violation fine is replaced by an automated process which reduces the overhead and turnaround time. The fines can be viewed and paid on-line.

1.5.4 Comparison

The differences between those types of text documents are due to their different creation and storage conditions, and to their different usage and goals. This comparison aims at helping us to understand why some approaches would succeed when applied to some kind of text document and fail with another one. It aims also at pointing out the important properties that have to be considered when trying to adapt an approach already used for a given kind of text document to another one.

Printed versus handwritten text

The height of the printed characters is more or less stable within a textline. On the other hand, the distribution of the height of handwritten characters is quite diverse. This is also the structural properties that help humans discriminate printed from handwritten text. These remarks stand also for the height of the main body of the characters as well as the height of both ascenders and descenders. Thus, the ratio of ascenders' height to main body's height and the ratio of descenders' height to main body's height would be stable in printed text and variable in handwriting. These properties were used by E. Kavallieratou and S. Stamatatos [KS04] to extract a vector feature that would help to classify a text region on printed or handwritten text. In fact, both machine-printed and handwritten text characters are often met in application forms, question papers, mail as well as notes, corrections and instructions in printed documents. In all mentioned cases it is crucial to detect, distinguish and process differently the areas of handwritten and printed text for obvious reasons such as: (a) retrieval of important information (e.g., identification of handwriting in application forms), (b) removal of unnecessary information (e.g., removal of handwritten notes from official documents), and (c) application of different recognition algorithms in each case.

Printed and Handwritten text versus image text

Compared to printed or handwritten text, image and video text can have a very bad resolution which ranges from 160x100 pixels to 720x480 pixels. For example, in CIF format (384x288 pixels) the size of a character is smaller than 10 pixels whereas for numerical documents, it is between 50 and 70 pixels. Moreover it has complex background, and artifacts due to compression. The reader can experiment by himself this fact by trying to feed an OCR with a text image (there are some free OCR software on the internet, for example, "SimpleOCR"), he will

receive the message: "can not convert this page" even with the cleanest images.

1.6 Contribution of this dissertation and outlines

The main contributions of this thesis are:

- The design of a new method for text image binarization, that we call the Convolutional Text Binarizer (CTB). This method is based on a new convolutional neural network architecture that processes color text images to directly produce binary text images using conjointly color distribution and the geometrical properties of characters. It insures robustness to noise, to complex backgrounds and to luminance variations. The details of the new architecture proposed as well as the experiments are exposed in chapter 4.

- The design of a new method for color text image segmentation. This method is also based on a new convolutional neural network architecture that processes color text images. For each processed image, this method produces a vector which indicates the frontier positions between consecutive characters. The details of the new architecture proposed as well as the experiments are exposed in chapter 5.

- The evaluation of the performance of a classical convolution neural network designed for isolated character image recognition on the public dataset ICDAR 2003. The details of the architecture proposed as well as the experiments are exposed in chapter 6.

- A discussion about the relevance of the binarization step in video text recognition based on machine learning. This study is exposed in chapter 7.

- The design of a new method for color text image recognition based on graph theory, that we call the image Text Recognition Graph (iTRG). This method uses from the text segmentation module and the character recognition module exposed in chapters 4 and 5. The details relative to the structure of the iTRG as well as the experimental results are exposed in chapter 8.

In addition, many experiments are achieved on the public ICDAR 2003 datasets in order to contribute to the clear understanding of the state of the arts.

Before continuing the exploration of the different new schemes proposed through this work, we report in the next chapter an overview of the current state of the art in video text recognition and we focus in chapter 3 on the Convolution Neural Networks which are the foundation of most modules proposed here. We finish this work by chapter 9, where we draw conclusions and perspectives for future directions.

Chapter 2

State of the art

Les hommes construisent trop de murs
et pas assez de ponts.

We build too many walls
and not enough bridges.
[Isaac Newton] (1642--1727)

2.1 Introduction

Most people learn to read and write during their childhood. They have no problem in reading fancy font styles, characters with missing or fragmented parts, words with multicolored characters, misspelled words, artistic and figurative designs, ...

On the contrary, despite more than five decades of intensive research, the reading skill of computers is still away behind that of human beings. This is especially true for reading text in images and videos. However, thanks to the recent advances in computing power, text recognition techniques also advanced notably. We present in this chapter, a quick overview of the state of the art techniques in the main modules of image text recognition.

We present text binarization in section 2.2, text segmentation in section 2.3 and character recognition in section 2.4. Conclusions are drawn in section 2.5.

The figure 2.1 shows the different categories of techniques and the relations between the different modules.

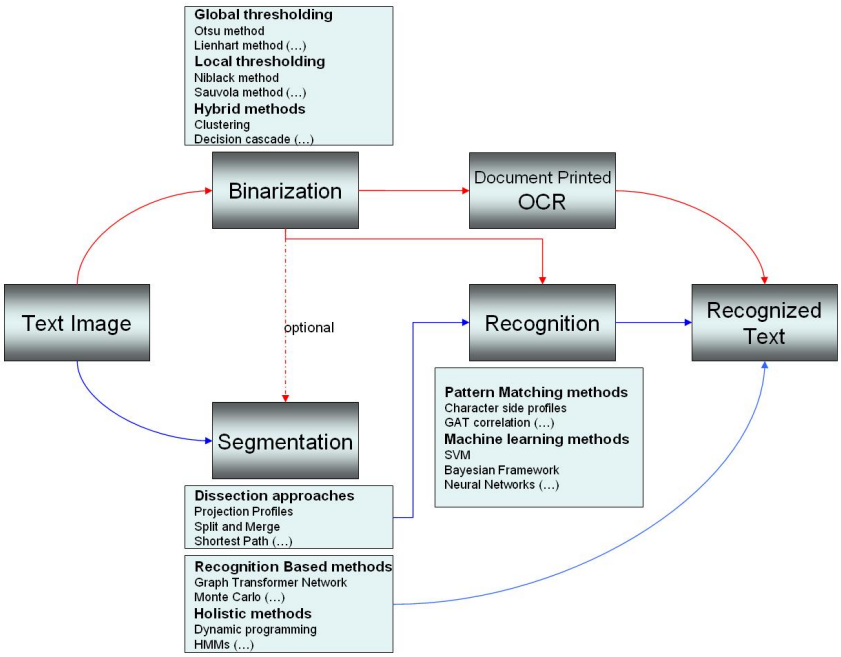


Figure 2.1: Video text recognition methods

2.2 Binarization

Image binarization converts an image of up to 256 gray levels to a black and white image. Usually, binarization is used as a pre-processing step before classical OCR in order to seperate the text pixels from the background pixels. In fact, most OCR packages on the market work only on bi-level images. But binarization can be used also as a pre-processing step before more sophisticated video text recognition systems. The simplest way to get an image binarized is to choose a threshold value, and classify all pixels with values above this threshold as white, and all other pixels as black. The problem then is how to select the correct threshold.

Basically, there are three categories of methods, the global thresholding, the local or adaptive thresholding and the hybrid methods.

2.2.1 Global thresholding

In the global thresholding methods, there is only one threshold for the entire image. Otsu's method [Ots79] is perhaps the best known and most widely used thresholding technique. Otsu's method assumes that images have two

normal distributions with similar variances. The threshold is selected by partitioning the image pixels into two classes at the gray level that maximizes the between-class scatter measurement. Let the pixels of a given image be represented in L gray levels $1, 2, \ldots, L$. The number of pixels at level i is denoted by n_i, and the total number of pixels by $N = n_1 + n_2 + \ldots + n_i + \ldots$. Then suppose that the pixels were dichotomized into two classes C_0 and C_1, which denote pixels with levels $1, .., k$ and $k+1, \ldots, L$, respectively. This method search for the gray level k which maximizes the between-class variance. The between class variance is defined as:

$$\sigma_B^2 = \omega_0 \omega_1 \left(\mu_1 - \mu_0\right)^2 \tag{2.1}$$

Where:

- $\omega_0 = \sum_{i=1}^{k}(P(i))$ is the frequency or the probability of the first class occurrence and $P(i) = n_i/N$ is the statistical frequency distribution of level i;
- $w_1 = 1 - w_0$ is the frequency or the probability of the second class occurrence;
- $\mu_0 = \frac{\sum_{i=1}^{k}(iP(i))}{w_0}$ is the mean of the first class;
- $\mu_1 = \frac{\sum_{i=k+1}^{L}(iP(i))}{w_1}$ is the mean of the second class.

Otsu's method yields good binary results. However, if the text intensities are inseparable from the background intensities, Otsu's method finds an unsuitable threshold value. One natural way to deal with this problem is to subdivide the image into sub-images and to apply various techniques to them. Lienhart et al [LW02] perform some pre-processing operations such as complex background removal then choose the intensity value halfway between the intensity of the text color and the background color as a good threshold. Estimating the intensity of the text and the background colors is based on the histogram distribution of the two upper and lower rows of the text box (as they should contain mainly background pixels), and on the histogram distribution of the four central raw of the text box (as they should contain mainly text pixels). They use their binarization method with a classical OCR and report an overall recognition rate of 69.9% on a large set of video sequences.

2.2.2 Local thresholding

Local thresholding methods compute a threshold for each pixel on the basis of the information contained in a neighborhood of the pixel. Most of the

methods calculate a threshold surface over the entire image. If a pixel (x, y) in the input image has a higher gray level than the threshold surface evaluated at (x, y), then the pixel (x, y) is set to white, otherwise it is set to black. Some methods in this class are histogram-based, such as the method of Kopf et al. [KHE05b], which after some pre-processing, estimate the dominant text colors based on histograms computed on eight-bit YUV images (four luminance bits, two bits for each chrominance component). Bins in the histogram of the text region that are larger than the corresponding bin in the adjacent region are defined as text color. Finally, to produce the binary image, they analyze each block between two separators and classify the pixel as text or background based on a region-growing algorithm and the colors previously determined.

However, usually in this class, the approaches are window-based, which means that the local threshold for a pixel (x, y) is computed from the gray values of the pixels in a window centered at (x, y). Many researchers proposed various formulas to compute the local threshold based on the minimum and the maximum gray values in each window [Ber86], or based on the mean and the standard deviation [Nib86] as follows: $T = m + k.s$, where T is the threshold for the central pixel of a rectangular window which is shifted across the image, m the mean, s the variance of the gray values in the window and k a constant. One drawback of this approach is noise created in areas which do not contain any text, due to the fact that a threshold is created in these cases as well.

A similar method which solves this problem is proposed in [SSHP97]. The authors add a hypothesis on the gray values of text and background pixels: text pixels are assumed to have gray values near 0 and background pixels are assumed to have gray values near 255. This hypothesis results in the following formula for the threshold: $T = m \times (1 - k \times (1 - s/R))$, where R is the dynamic range of the standard deviation. This method gives better results for document images than [Nib86], but creates additional problems for video frames whose contents do not always correspond to the hypothesis, even if images containing bright text on dark background are reversed to resemble better the desired configuration. A comparative study of this kind of methods can be found in [HDDK05]. The performance of all these algorithms depends on the window size. The suitable window size for a document image is heavily affected by the character size as well as the character thickness. If the window size is too large, the details of the character structure are degraded in many cases. On the other hand, if the window size is too small, compared with the character thickness, the window centered at an inner pixel produces a large mean and a small standard deviation, because most pixels in that window belong to the foreground. As a consequence, the local threshold for such a

pixel is likely to be assigned a large value, and therefore, the central areas of thick and black regions were often classified into background. In order to avoid the above problems, Kim [Kim04] proposed a scheme based on three windows. The first window is used for a preliminary binarization for the analysis of the document image. The second and the third windows are used to determine the actual local thresholds. The overall flow of the algorithm is as follows: firstly, the input image is binarized with a moderate-size window. With the preliminary binarization result, text lines are extracted and their properties are analyzed. Then, for each text line, the system determines the sizes of the second and the third windows according to the analysis result. The size of the second window is assigned a large enough value to reflect the global trend; in general, it is the average height of the characters which is chosen as the size of the second window. On the other hand, the size of the third window is assigned a small enough value to catch on the detail of the character structure; a size comparable to the character thickness is a good choice. Finally, each text line is binarized with both the second and the third windows to produce the final result. The author uses with this scheme a commercial numeral recognizer software, named *'Inzi i-Form numeral recognizer'* and reports a recognition rate of 99.35% on good images of phone number cards and 96.71% for bad images of phone number cards.

2.2.3 Hybrid methods

Decision cascade method

In the case of local thresholding techniques, deciding what to do with each possible sub-image requires the establishment of a set of rules and determination of the values of a set of parameters. For this purpose, Chou et al [CHLC05] use a decision tree approach. Decision tree methods have been applied extensively to classification problems to identify the class types of input objects. The advantage of this approach lies in its ability to learn the order of decision rules and the values of parameters from training samples. In this case, each training sample consists of a feature vector and an associated action for each sub image. The feature vector consists of three components:

- Feature1: The difference between the Otsu threshold and the minimum Otsu threshold corresponding to the sub image.
- Feature2: The mean of the sub image.
- Feature3: The standard deviation of the sub image.

The set of possible actions are the following:

- Action1: local threshold = 0
- Action2: local threshold = 255
- Action3: local threshold = Otsu threshold for the processed sub-image
- Action4: local threshold = the minimum Otsu threshold for the processed sub-image

The framework is a decision cascade in which the decision node D_i has one branch leading to the action node A_i and another branch leading to the next decision node D_{i+1}, with $i = 1, 2, \ldots, n-1$. Meanwhile, the last decision node D_n has branches leading to action nodes A_n and A_{n+1}. In the decision node D_i, a pair (F_i, B_i) is specified, where F_i is a feature type and B_i is a branch point. The decision is made in such a way that when the F_i value of an input object O falls above or below B_i, action A_i is taken on O; otherwise, O is passed to the next decision node, or to the final action node, if $i = n$. Using an English OCR software system for recognition, the authors reported for their method, a recall rate of 96.02% and a precision rate of 95.22%.

Clustering methods

The goal here is to avoid the use of global thresholds or local thresholds which are difficult to determine and depends a lot on the image processed. The binarization is considered here as a clustering scheme where the system has to classify each region on text or non text region. Then most of the time, to evaluate the binarization performance, the result is fed to a commercial OCR system which will decide if the parameters of the clustering algorithm needs to be adjusted or not. Gllavata et al [GESF04] have used this method. First, they enhance the image resolution by cubic interpolation. Afterwards, they determine the color of the text and that of the background based on a color quantizer. In fact, the number of colors in the text box is reduced to the most dominating colors using a color quantization method. Then, two color histograms are calculated: the histogram of the central row in the text box and the histogram of the rows directly above and underneath of the text box. Finally, the difference histogram of those two histograms is calculated. The text and the background color are defined as the maximum and the minimum of this difference histogram. Once the text color is determined, this feature will be used in addition to the standard deviation of a wavelet transformed image to distinguish between text and non-text pixels. To perform this classification, a slightly modified k-means algorithm is applied which is used to produce a binarized text image. The segmentation result is fed into commercial OCR software to assess the binarization quality. The authors

report a recognition rate of 85.9% tested on 20 images with good resolution and 70.7% tested on 20 images with low resolution.

In a similar approach, Yokobayashi et al. have proposed two binarization techniques. In the first one [YW05], they choose one of the (Cyan, Magenta, Yellow) color planes using the maximum breadth of histogram to perform color clustering. In the second one [YW06], they perform color clustering based on the maximum between class separability.

Some techniques quantify the colors into a reduced number of dominant colors based on clustering algorithms, such as K-means. In [GA00] , this number is fixed to four. Then, the color or the two colors with the biggest rate of occurrences in the image border areas are considered as the background colors. The remaining colors are to be assigned to text or background. All the combinations are considered, each of which results in a possible binary image. The best one among them is determined by analyzing the periodicity of the vertical profiles of the binary image. This kind of methods depends on a lot of parameters, usually difficult to determine (the number of quantification levels, the size of the image border, thresholds of the periodicity,...).

Markov models methods

In some works, the binarization of characters was modeled as a Markov random field (MRF), where the randomness is used to describe the uncertainty in pixel label assignment. This technique was used in [CH97] to binarize license plate images, in [HSAAEF06] to binarize scanned document and in [WD02] to binarize low quality text images. In the latter work, the binarization system creates a Markov random field model which parameters are learned from text within training data and uses it in a Bayesian estimation algorithm to binarize noisy text images. The authors evaluate their method with the commercial OCR abby Finereader and report a recognition rate of 82% when tested on pink panther database and the university of Washington collection.

2.3 Segmentation

Text segmentation is sometimes confused with text binarization in the literature; in this report, we call text segmentation, the step aiming at segmenting the text into symbols (characters, syllabus, words...) identifiable by the recognizer. This task is still receiving a great interest because of its crucial impact on the performance of the overall recognition system. Thus there is a wide range of methods. We classify them as in [CL96] into three categories. The first one concerns the classical approaches, where character regions are identified based on their properties, called here dissection approaches. The

second one is the recognition based segmentation also known as segmentation free methods, because no complex cutting process is used but rather a decision on the character region based on the recognition result. The third one tries to recognize words as a whole avoiding character segmentation and thus named holistic methods. Of course, there are hybrid methods which are somehow a combination of the others.

2.3.1 Dissection approaches

Methods discussed in this section are based on the decomposition of the image into a sequence of regions using general features. This is opposed to later methods that divide the image into sub-images independently from content. Dissection is an intelligent process in that an analysis of the image is carried out.

Projection profiles

This technique belongs to the first employed techniques for text segmentation; it is applied to a binary image. The vertical projection analysis consists of a simple running count of the black pixels in each column so that it can detect the white space between successive letters. Some functions were used to improve this method, such as the peak-to-valley function. In this function, a minimum of the projection is located and the projection value noted. The sum of the differences between this minimum value and the peaks on each side is calculated. The ratio of the sum to the minimum value itself (plus 1, presumably to avoid division by zero) is the discriminator used to select segmentation boundaries. This ratio exhibits a preference for low valley with high peaks on both sides. This technique is mostly used for machine printed character segmentation, where the image resolution is high and the characters are usually well separated, whereas in text images, usually it is either used for detection and localization [KHE05b],[LW02],[GEF04],[YZHT03] or as a simple first step in the recognition based segmentation methods [SKHS98].

Split and merge algorithm

The Split and Merge algorithm was first proposed by Horowitz and Pavlidis [HP76]. It is based on a hierarchical decomposition of the frame. According to Horowitz and Pavlidis, the split process begins with the entire image as the initial segment, which is then split into quarters. Each quarter is tested against a certain homogeneity criterion to determine whether the segment is "homogeneous enough". If not homogeneous enough, the segment is split again into quarters. This process is applied recursively until only homogeneous segments are left. The standard homogeneity criterion used in the case of text image segmentation is based on the assumed mono-chromaticity of characters, which means that the difference between the largest and smallest

gray tone intensities is supposed to be lower than a certain threshold in the case of a character region. Next, in the merge process, adjacent segments are merged together if their average gray tone intensity difference is less than a given threshold. As a result, all monochrome characters appearing in the image should be contained in some of the monochrome segments. The segmented image now consists of regions homogeneous according to their gray tone intensity. Some regions are too large and others are too small to be instances of characters. Therefore, some post processing is required. In [Lie96] monochrome segments whose width and height exceed a predefined maximum size are removed, as are connected monochrome segments whose combined expansion is less than a predefined minimum size. Similarly, the contrast, the fill factor and the aspect ratio are checked, and if one of them exceeds some limits, the region is discarded. The segmentation performance of this process ranges from 86% to 100%. In [KA04], the authors used also a split and merge algorithm for text web images segmentation based on the human color perception properties. First of all, they perform a conversion of the RGB data stored in the image file into the HLS representation. In fact, the HLS color space is chosen since the factors that enable humans to perform (chromatic) color differentiation are mainly the wavelength separation between colors (expressed by Hue differences), the color purity of the involved colors (expressed by Saturation) and their perceived luminance (expressed by Lightness). Thus, there are two homogeneity criterions used in the split and merge algorithm, the first one is based on the histogram of the lightness and the second one based on the histogram of hue. For each histogram, peaks are identified then analyzed. The splitting and merging process are then conducted depending on whether adjacent peaks are perceived to be similar by a human observer or not. In the merging process, connected component analysis is used and the desired result at the end would be that characters in the image are described by connected components not containing parts of the background. To refine the obtained result, text line extraction technique is used based on size constraints and Hough transform.

Shortest path method

A shortest-path method produces an optimum segmentation path using dynamic programming. This method was proposed as robust character segmentation for low-resolution images, by Kopf et al. [KHE05b]. They propose a three-step approach. First, enhance the resolution by linear interpolation to scale the text region by a factor of four. Then locate separators for individual characters. For this purpose, they assume that usually, the contrast between text pixels and background pixels is high, whereas the average difference between adjacent background pixels is much lower. Therefore they search a path from the top to the bottom of the text region. Different start-

ing positions in the top row are selected, and the paths with the lowest costs are stored. The costs of a path are defined as the summarized pixel differences between adjacent path pixels. The path with the minimum cost which is called cheapest path rarely crosses character pixels and defines a good separator for characters. They use the Dijkstra shortest-path algorithm for graphs to identify the separators. Each pixel is defined as a node that is connected to three neighbor pixels (left, right, down). The costs of travel from one node to another are defined as the absolute luminance differences between these two pixels. They start at different positions in the top row and calculate the path from each position to the bottom row. A major advantage of this approach is the fact that no threshold is required to locate the separators of characters. The authors report a segmentation error rate of 9.2% from which 3.8% are due to characters split and 5.4% are due characters merged.

Following the same idea, Jia et al. [TJCY07] propose also a character segmentation method using shortest path in document images. They first find the connected components of the binary word image. Next, the connected components are analyzed to determine whether or not each one of them consists of one or more characters. If the connected component consists of one or more characters, the correct segmentation path is found using a shortest path approach based also on the Dijkstra algorithm. The authors report a segmentation accuracy of 95% over a set of about 2,000 smeared characters.

2.3.2 The recognition based methods

In order to avoid choosing thresholds to segment video text characters, which is in general a difficult task since these parameters depends a lot on the text characteristics, such as size, thickness, and also on the complexity of the background, and the quality of the image, some works concentrated on recognition based segmentation also known as segmentation free methods, because no complex cutting process is used but rather a decision on the character region based on the recognition result.

For example, in [SKHS98] the authors use a recognition-based segmentation method on videos. They first enhance the image which will be processed by sub-pixel interpolation and multi-frame integration, and then based on the observation that a character consists of four different directional line elements: 0° (vertical), 90° (horizontal), 45° (anti-diagonal), and 45° (diagonal), four specialized character extraction filters are applied. Afterwards, thresholding at a fixed value for the output of the character extraction filter produces a binary image which is used to determine positions of characters.

Projection profiles give a first character segmentation which will be refined by the character recognition process. This process is based on a simple matching technique between reference patterns and the character segment. This method almost doubles the recognition rate of a conventional OCR technique that performs binarization of an image based on a simple thresholding, character extraction using a projection analysis, and matching by correlation.

In [BMLC+92], the authors combine dynamic programming and neural network recognizer for segmenting and recognizing character strings. First, some preprocessing is applied to the image to remove noise, such as removing underlines, deslanting and deskewing the image. Then, they use modulated gradient hit to generate cuts. Afterwards, they associate with each segment a node in a graph and connect nodes with arcs such that two nodes are connected if and only if they represent neighbor segments. Finally, each node is sent to the recognizer for classification and scoring. The recognizer which is based on neural networks is trained so that it learns the segmentation error and become able to select the good nodes corresponding to the correctly segmented characters.

This method was improved by LeCun et al. [LBBH98]. The authors use a graph transformer network instead of the neural network. Graph Transformers are modules which take one or several graphs as input and produce graphs as output. A network of these modules is called Graph Transformer Network (GTN). While conventional network use a fixed size vectors, each arc in the graph contains a vector, providing the network with more size flexibility. From the statistical point of view, the fixed-size state vectors of conventional networks can be seen as representing the means of distributions in state space. In GTN's the states are represented as graphs, which can be seen as representing mixtures of probability distributions over structured collections or sequences of vectors. However, their scheme is used for handwritten characters, which do not have the same characteristics with video text characters.

Chen et al [CO03] proposed a segmentation method named the Monte Carlo Video Text Segmentation (MCVTS). This method presents a probabilistic algorithm for segmenting and recognizing text embedded in video sequences based on adaptive thresholding using a Bayes filtering method. The key point of this method is to represent the posteriori distribution of text threshold pairs given the image data by a set of weighted random samples, referred to as particles. This randomness allows adapting to changes of grayscale values. Unlike other filtering techniques, this method allows to evaluating the likelihood of the segmentation parameters directly from the corresponding recognized text string based on language modeling and OCR statistics. The authors report a segmentation recall rate of 93.9% and a seg-

mentation precision rate of 85.3% when the system is tested on a database of 230 sequences of text images that are tracked and extracted from news videos and 17 sequences of text images from documentary videos.

2.3.3 Holistic methods

Given that character segmentation is a difficult task which is most of the time the major factor in the recognition errors, some works try to avoid this step by applying segmentation and then recognition directly on words instead of characters. These methods are also considered sometimes as segmentation free approaches since there is no character segmentation as mentioned above.

Whole word recognition was introduced by Earnest at the beginning of the 1960s [Ear62]. In this work, Recognition was based on the comparison of a collection of simple features extracted from the whole word against a lexicon of "codes" representing the "theoretical" shape of the possible words. Recent holistic methods are still based on this idea of firstly, performing feature extraction and secondly, performing global recognition by comparing the representation of the unknown word with those of the references stored in the lexicon.

Chen and Decurtin [CdC93] have proposed a word recognition method for classical OCR systems that consists in four steps: (1) feature extraction, (2) matching, (3) weighted voting, and (4) word recognition. They use diverse types of contour and skeleton features. Two features are considered matched if the difference in every attribute between the two is less than a predefined tolerance. When a detected feature matches a model feature of a reference character, the character will receive one vote from that feature. To take into account the utility of the features that voted for the character, the authors proposed a weighted voting based on the detectability defined as the probability of detecting the feature when the character is present in the image and the uniqueness defined as the probability that a detected instance of the feature is actually caused by the character. Finally, words candidate are selected from the lexicon and their corresponding confidence rates are evaluated based on the confidence of its constituent characters. Manual evaluation shows that approximately 80% of the words were correctly recognized.

Some word candidates comparison techniques are inspired from works in speech recognition such as Dynamic Programming [MPBP91], [PSH+93] with optimization criteria based either on distance measurements such as the Edit Distance or on a probabilistic framework such as Markov or Hidden Markov Models [NWF86], [BGL93]. Most of these works concern handwriting recognition.

A major drawback of this class of methods is that their use is usually restricted to a predefined lexicon: since they do not deal directly with letters but only with words, recognition is necessarily constrained to a specific lexicon of words. This point is especially critical when training on word samples is required because, each time that the lexicon is modified or extended with new words, a new training stage is necessary to update the system. This property makes this kind of method more suitable for applications where the lexicon is statically defined (and not likely to change), like check recognition. They can also be used for on-line recognition on a personal computer (or notepad), the recognition algorithm being then tuned to the writing of a specific user as well as to the particular vocabulary concerned.

2.4 Character Recognition

In the previous section 2.3, we presented the segmentation step. We saw that in many cases, the outputs of the segmentation step are probable character images that need to be recognized. Character recognition is then the final step and should return what is the character in the processed image after segmentation and/or binarization. Even though many works use available OCR systems designed conventionally for printed text after some pre-processing steps [HYZ02],[MH01], we will focus here only on systems designed for image character recognition. We can classify the character recognition methods in two categories: pattern matching and pattern classification.

2.4.1 Pattern matching methods

This is an old principle that was proposed for OCR in 1929 by Tausheck. It reflects the technology at that time, which used optical and mechanical template matching. Light passed through mechanical masks is captured by a photo-detector and is scanned mechanically. When an exact match occurs, light fails to reach the detector and so the machine recognizes the characters printed on the paper. Nowadays, the idea of template matching is still used but with more sophisticated techniques. A database of models is created and matching is performed based on a distance measure. The models are generally composed of specific features that depend on the properties of the pattern to be recognized.

Character Side Profile

Chen at al [CGR04] proposed a method based on character side profiles, in videos. First, a database is constructed with left, right, top and bottom side-profiles of sample characters. Then the candidate characters

are recognized by matching their side-profiles against the database. This method requires of course a 'cleaned' binary text image. Therefore, they apply various pre-processing techniques before the recognition step, namely: shot boundary detection, edge-based text segmentation, multiple frame integration, gray-scale filtering, entropy-based thresholding, and noise removal using line adjacency graphs (LAGs). The authors reported a character recognition rate varying from 74.6% to 86.5% according to the video type (sport video, news, commercial videos...).

Curvature Scale Space (CSS)

Another template matching method was proposed by kopf et al. [KHE05b]. They have chosen to analyze the contour of a character and derive features extracted from the curvature scale space (CSS). This technique which is based on the idea of curve evolution requires also binary text images. A CSS image is defined by the set of points where the curvature is null. In many cases, the peaks in the CSS image provide a robust and compact representation of a contour with concave segments. For characters without concave segments (e.g. 'I' and 'O'), the authors proposed the extended CSS method, where the original contour is mapped to a new contour with an inverted curvature, thus, convex segments become concave and the problem is solved.

The matching process is done by comparing the feature vectors (CSS peaks) of an unknown character to those of the characters that are stored in a database. It might be necessary to slightly rotate one of the CSS images to best align the peaks. In fact, shifting the CSS image left or right corresponds to a rotation of the original object in the image. Each character is stored only once in the database, and for instance, the horizontal moves compensate small rotations of italic character.

If a matching peak is found, the Euclidean distance of the height and position of each peak is calculated and added to the difference between the CSS images. Otherwise, the height of the peak in the first image is multiplied by a penalty factor and is added to the total difference. If a matching is not possible, the two objects are significantly different. This rejection helps to improve the overall results because noise or incorrectly segmented characters are rejected in the matching step. A recognition rate of 75.6% is reported for a test set of 2986 characters extracted from 20 artificial text images with complex background.

Gaussian Affine Transform (GAT) correlation

Yokobayashi et al [YW05, YW06] proposed two systems for character recognition in natural scene images. Both of them rely on two steps: the first one is the binarization step and the second one is the recognition step based on an improved version of GAT correlation technique for grayscale character images.

In [YW05], the authors proposed a local binarization method applied to one of the Cyan/Magenta/Yellow color planes using the maximum breadth of histogram. This idea is based on the hypothesis that the more information entropy of grayscale occurrence a given image has the more effectively and easily a threshold value of binarization for the image can be determined, given that the breadth of grayscale histogram is closely related to the information entropy contained in the color plane. Therefore, they apply local binarization to the selected color plane. They compute the mean value of gray levels of all pixels in the selected color plane. Then, if a majority of nine pixels in a 3×3 local window have smaller gray levels than this mean, the pixel is considered as a character pixel, otherwise it is considered as a background pixel.

Once a binary image is obtained, an improved GAT correlation method [WKT01] is applied for recognition. This is a matching measure between the binary character image and a template image. As templates, the authors use a single-font set of binary images of alphabets and numerals, which explains the need for the previously mentioned binarization step.

To obtain a measure that is robust to scale change, rotation, and possible distortion, the correlation should be computed on transformed images. Therefore, the authors search for optimal affine transformation components, which maximize the value of normalized cross-correlation, by using an objective function based on a Gaussian kernel. Once these parameters are determined, they compute the correlation between the transformed input image and a template image. Then, they compute the correlation between the input image and the transformed template image, and finally the average of these two values is used as the match score. The authors report an average recognition rate of 70.3%, ranging from 95.5% for clear images to 24.3% for little contrast images, from the ICDAR 2003 robust OCR sample dataset.

In [YW06], the authors proposed a binarization method based on three steps. Firstly, color vectors of all pixels in an input image are projected onto different arbitrarily chosen axis. Secondly, they calculate a maximum between-class separability by setting an optimal threshold according to the Otsu's binarization technique [Ots79]. Thirdly, they select the axis that gives the largest between-class separability and the corresponding threshold for binarization of the input image. Then, they decide which class corresponds to characters or background according to the ratio of black pixels to white ones on the border of the binarized image. As in their previous work [YW05], once the binary image is obtained, an improved GAT correlation method is applied for recognition. The authors report an average recognition rate of 81.4%, ranging from 94.5% for clear images to 39.3% for seriously distorted images, from the ICDAR 2003 robust OCR sample dataset.

2.4.2 Machine learning methods

Actually, this category of techniques is more used in handwritten and document images character recognition [LBBH98], [MGS05], [Nag00] than in video text recognition.

Support vector machine

In [DAS01], a wide range of feature vectors are extracted: vector template features, regional features, balance and symmetry features, skeleton features, corner features.

The total feature vector for each character contains 172 elements and is presented to the classifier as input. The character classifier is based on support vector machine which are trained to learn the mapping from this 172-dimensional input feature space to an output recognition space consisting of 62 classes corresponding to 26 lower case letters, 26 upper case letters, and 10 digits. The authors report a recognition rate of 87% on subtitles, 72% on scrolling credits and 68% on news captions.

Bayesian framework

In [ZC03] a Bayesian framework is used for word recognition. First, a feature set for character recognition including Zernike magnitude, Direction proportion, 1st-order peripheral features, second-order peripheral features, vertical projection profile and horizontal projection profile is build. Then the feature vector of a given word is constructed based on its characters feature sets. The videotext word recognition problem can be formulated using Bayesian method or the maximum a posterior (MAP) recognition which is based on the construction of a language model based on a training set and a word observation model based on the corresponding word feature vector. The authors report a word recognition rate of 76.8% and character recognition rate of 86.7%.

Neural networks

Jacobs et al [JSVR05] use convolutional neural networks for character recognition in low resolution images. This recognizer takes a 29×29 window of image data as input, and produces a vector containing a probability distribution over the set of characters in the alphabet. This vector contains, for each character, the probability that the input window contains an image of that character. For the alphabetic characters, the character recognizer has an 87% success rate. While, for the alphanumeric characters, it drops down to 83% success. When all of the punctuation is included, the accuracy drops to 68%. When this system is combined with a language model using dynamic programming, the overall performance is in the vicinity of 80-95% word accuracy on pages captured with a 1024×768 webcam and 10-point text.

In [LLM06] a structure called Hierarchical Radial Basis Function (RBF) Neural Networks is proposed to recognize particular characters contained in text images. The networks involve several sub-RBF neural networks connected like a tree, which could classify characters. Particle Swarm Optimization, a newly developed global search evolutionary algorithm, is enrolled to train this structure. The authors report only the performance on training data and not on a test set.

2.5 Conclusion

We have reviewed in this section some of the main techniques used in the different modules of an image text recognition system: binarization, segmentation and recognition. We notice that many methods are based on some statistical analysis of the luminance or chrominance distribution generally based on histograms, without taking into account the characters shapes. In addition, most of them are assuming to start from binary images. This is due probably to the inherited techniques from document printed OCR. In fact, video OCR is considered sometimes as an extension of classical OCR, while we believe that it is a totally different problem with its own properties and challenges. Moreover, only few works tested their methods on public datasets. Actually, there is only one public dataset, namely the ICDAR 2003 robust reading dataset which is really difficult due to highly distorted text images.

Throughout this work, we propose machine learning techniques which take into account the geometrical properties of characters in addition to the luminance or chrominance distributions. We pay attention to test our methods on the public dataset cited previously in order to contribute to the clear understanding of the state of the art. We also study the actual relevance of binarization in our scheme.

Chapter 3

Convolutional Neural Networks

Les détails font la perfection, et la perfection n'est pas un détail.

Details make the perfection, and the perfection is not a detail.
[Léonard de Vinci] (1452--1519)

3.1 Introduction to Neural Networks

An artificial neural network (ANN), also called a simulated neural network (SNN) or simply a neural network (NN), is an interconnected group of artificial nodes (called variously neurons, neurodes, PEs (processing elements) or units) which are connected together to form a network of nodes - hence the term neural network.

The original inspiration for the technique was from examination of bioelectrical networks in the brain formed by neurons and their synapses. So similarly, a basic neuronal model called perceptron has a set of connecting weights, a summing unit, and an activation function as shown in figure 3.1. The activation function can be either linear or nonlinear. A popular nonlinear function used in NNs is the sigmoid which is a monotonically increasing function that asymptotes at some finite value as $\pm\infty$ is approached. The most common examples are the standard logistic function $f(x) = \frac{1}{1+e^{-x}}$ and the hyperbolic tangent $f(x) = \tanh(x)$ shown in figure 3.2.

The sigmoid function has the advantage of being derivable so that gradient learning is possible as we will see later on. Moreover the use of sigmoid enables the probabilistic interpretation of the neuronal outputs. In fact, if we consider two classes having Gaussian density functions $P(\omega_1)$ and $P(\omega_2)$; and with a common covariance matrix Σ, the a posterior probability is a

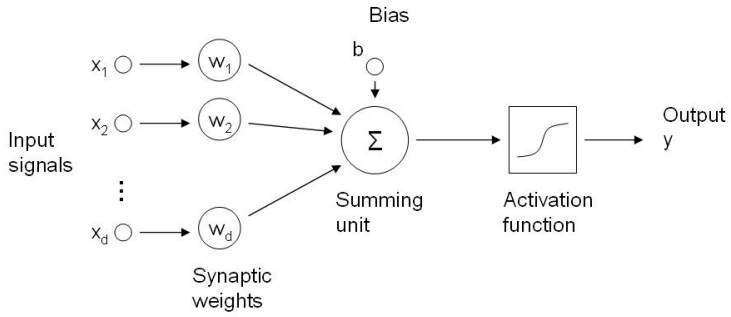


Figure 3.1: Model of neuron

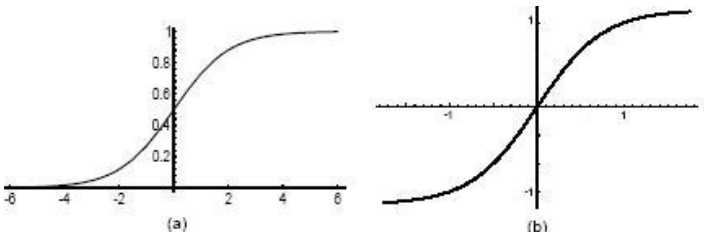


Figure 3.2: Sigmoid functions: (a) Logistic, (b) Hyperbolic tangent

sigmoid function as shown in the following equation:

$$\begin{aligned} P(\omega_1|x) &= \frac{P(\omega_1)p(x|\omega_1)}{P(\omega_1)p(x|\omega_1)+P(\omega_2)p(x|\omega_2)} \\ &= \frac{P(\omega_1)e^{[\frac{(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)}{2}]}}{P(\omega_1)e^{[\frac{(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)}{2}]}+P(\omega_2)e^{[\frac{(x-\mu_2)^T\Sigma^{-1}(x-\mu_2)}{2}]}} \\ &= \frac{1}{1+\frac{P(\omega_2)}{P(\omega_1)}e^{-[(\mu_1-\mu_2)^T\Sigma^{-1}x+C]}} \\ &= \frac{1}{1+e^{w^Tx+b}} \end{aligned} \tag{3.1}$$

where, C is a constant independent from x, $w = \Sigma^{-1}(\mu_1 - \mu_2)$ and $b = C + log\left(\frac{P(\omega_1)}{P(\omega_2)}\right)$.

The basic model shown in figure 3.1 can be extended to the single layer network as shown in figure 3.3 by adding further output neurons, each corresponding to a class pattern.

In the case of linear activation function, the output $y_k(x)$ of a given neuron k is a linear discriminant function:

$$y_k(x) = \sum_{i=1}^{d} w_{i,k}x_i + w_{0,k} = \sum_{i=0}^{d} w_{i,k}x_i \tag{3.2}$$

where $(x_i)_{i=1..d}$ are the input signals, $w_{i,k}$ is the weight connecting the input

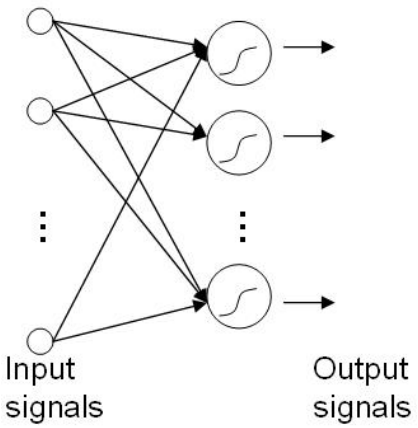


Figure 3.3: Single layer neural network.

node i to the output node k and $w_{0,k}$ is a bias term, which can be viewed as the weight corresponding to an input signal $x_0 = 1$.

The input pattern is classified to the class with the maximum output. Using a sigmoid activation function, which is monotonic, does not affect the decision of classification because the order remains unchanged. Hence this architecture is unable to separate classes with complicated distributions, for which the decision boundary is generally nonlinear such as the XOR (exclusive error) problem. A way to enhance the separation ability is to add at least on further layer, called hidden layer to the basic model. In fact, using nonlinear functions of pattern features as the inputs to linear combiner is a nonlinear function of the original features.

This architecture is called the feed-forward multi-layer perceptron (MLP) network, it is shown in figure 3.4. It was the breakthrough needed to pioneer current neural network technologies. In fact, the power of an MLP network with only one hidden layer is surprisingly large. Hornik *et al.* and Funahashi showed in 1989 [HSW89], [iF89], that such networks are capable of approximating any continuous function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ to any given accuracy, provided that sufficiently many hidden units are available.

MLP networks are typically used in supervised learning problems. This means that there is a training set of input-output pairs and the network must learn to model the dependency between them. The training here means adapting all the weights and biases to their optimal values. The criterion to be optimized is typically the mean square error (MSE).

The supervised learning problem of the MLP can be solved with the back-propagation algorithm [Wer90]. The algorithm consists of two steps. In the forward pass, the predicted outputs corresponding to the given inputs are

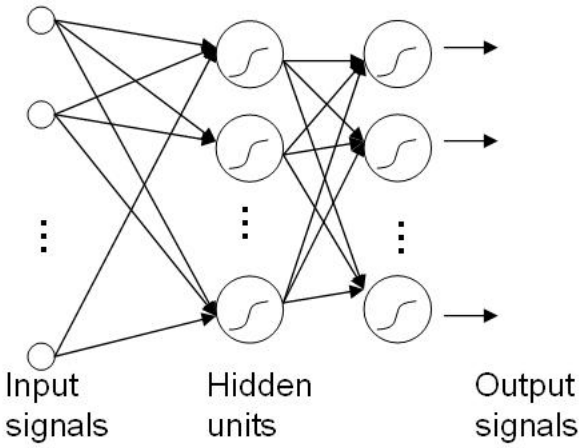


Figure 3.4: Multilayer perceptron

evaluated. In the backward pass, partial derivatives of the cost function with respect to the different parameters are propagated back through the network. The chain rule of differentiation gives very similar computational rules for the backward pass as the ones in the forward pass. The network weights can then be adapted using any gradient-based optimization algorithm. The whole process is iterated a predefined number of iteration or until the weights have converged.

Of course, in addition to the feed-forward MLP networks, there are many different kinds of NNs such as: Radial basis function (RBF) networks, Kohonen self-organizing networks, Recurrent networks, Stochastic neural networks, Modular neural networks,...

Moreover, neural networks have shown nowadays, high performance in a wide range of fields: finance (financial forecasting, market segmentation, real estate evaluation, ...), industry (robotics, automotive industry, plant modeling, ...), medicine (bio informatics, medical diagnosis, survival probability estimation, ...), new technologies (search engines, computer games, ...).

Intrigued by the capacity of neural networks, we focus in this work on NNs to tackle the problem of text recognition in complex images. However, while multilayer perceptron networks perform well for rather simple classification problems, they have several drawbacks when it comes to image processing applications. First, the number of trainable parameters becomes rapidly extremely large. For example, a 24×24 input layer would already have 576 connections per single neuron in the hidden layer. Secondly, it offers little or no invariance to shifting, scaling, and other forms of distortion. Third, the topology of the input data is completely ignored, yielding similar training

results for all permutations of the input vector.

That's why, we choose to concentrate our efforts on convolutional neural networks (CNNs) which are specific feed-forward multilayer neural networks with demonstrated results in handwritten character recognition [LBBH98] and face detection [GD04]. We present, in section 3.2, a brief history of CNNs. Then, we will detail in section 3.3 a typical architecture of CNNs and in section 3.4 the training methodology. In section 3.5, we describe some applications. Conclusions are drawn in section 3.6.

3.2 A Brief History

Warren S. McCulloch and Walter Pitts have proposed in the forties the very first neural network computing model. In the late fifties, Rosenblatt's work [Ros58] resulted in a two-layer network, which was capable of learning certain classifications by adjusting connection weights, but was not able to solve the classic XOR (exclusive or) problem which causes decline in interest towards neural network research. It is only on the eighties that interest in neural networks resurged with Boltzmann machines, Hopfield nets, competitive learning models, and multilayer networks which thanks to their hidden layers solve the XOR problem.

Convolutional Neural Networks were also proposed in the eighties as non linear image processing algorithms. The network topology is similar to the mammalian visual cortex [HW62] and is based on some of its characteristics such as the local receptive fields i.e. each neuron is only connected to a sub-region corresponding to a certain number of neighboring neurons, in the preceding layer and the cascade of structure of neurons.

In the litterature, there are two renowned convolutional neural networks that have been developed, namely the *Neocognitron* [Fuk80] and the *LeNet-5* [LBBH98].

The Neocognitron is the first convolutional neural network proposed in the eighties by Fukushima [Fuk80], [FMI83].

The neocognitron consists of layers of simple cells corresponding to feature extracting cells and called S-cells, and layers of complex cells corresponding to recognition cells and called C-cells. These layers of S- and C-cells are arranged alternately in a hierarchical manner. In other words, a number of modules, each of which consists of an S- and a C-cell layer, are connected in a cascade in the network. The network has forward and backward connections between cells. In this hierarchy, the forward signals manage the function of pattern recognition, while the backward signals manage the function of selective attention, pattern segmentation, and associative recall.

Some of the connections between cells are variable, and the network can acquire the ability to recognize patterns by unsupervised learning. After some years, Fukushima moved away from the self organization scheme in favor of a supervised learning algorithm in order to enhance the classification performance of the neural network for handwritten character recognition [FMI83]. Since then, Fukushima proposed many extensions to the basic model of the neocognitron [Fuk88], [FS98], [Fuk01].

In the nineties, LeCun and his colleagues [CBD+90] developed a series of convolutional neural networks, called LeNet (1-5). Their network architectures consist of a cascade of feature detection layers where instead of having an S-layer followed by a C-layer, the networks have alternative convolution and sub-sampling layers. Moreover, the model of the individual neurons is the basic perceptron with a sigmoid activation function which was not the case in the neocognitron. Similarly to the neocognitron, these convolutional neural networks are based on three architectural ideas:

1. local receptive fields which are inspired from the mammalian visual cortex and which are used to detect elementary visual features in images, such as oriented edges, end points or corners;

2. shared weights which allows to extract the same set of elementary features from the whole input image and to reduce the computational cost;

3. sub-sampling operations which reduce the computational cost and the sensitivity to affine transformations such as shifts and rotations.

An important difference between the Fukushima model and the LeCun model concerns the training procedure. In fact, LeCun *et al.* were the first to apply back-propagation to convolutional neural networks. Good performance of such model is demonstrated through the literature as many variants of it have been proposed (LeNet-5 [LBBH98] for handwritten character recognition, CFF [GD04] for face detection, Space Displacement Neural Networks [BLH93] for word recognition, Siamese CNNs [CHL05] for face verification, Shunting Inhibitory Convolutional Neural Networks [TB03] for face detection, Sparse Convolutional Neural Networks [Gep06] for object classification).

Usually in the previously cited models, the number of layers, the number of feature maps and their dimensions and the connection scheme are adapted to the given problem. Unfortunately, these parameters have to be determined experimentally. There is no algorithm, until now, able to automatically determine the optimal architecture of a CNN for a given classification task.

As the LeCun *et al.* model forms the basis of all the convolutional neural network architectures presented in this work we will describe it in more detail in the following sections.

3.3 Architecture overview

The architecture of LeCun *et al.* model corresponds to a feed-forward multilayer network. We detail here the most famous model called LeNet-5 [LBBH98]. It consists of seven layers, excluding the input layer. Each layer of the first five layers contains one or more planes, and each plane is made up of a 2D lattice of neurons. Each neuron is locally connected to a set of neurons located within its local receptive field in the planes of the previous layer. These planes are called feature maps.

The input layer E contains one plane $e^{(0)}$ of size 32×32 and receives the gray-level image containing the handwritten digit to be recognized. The pixel intensities are normalized between -1 and +1 because convergence is usually faster if the average of each input variable over the training set is close to zero [LBOM98].

The first hidden layer C_1 consists of four feature maps $(c^{(1,m)})_{m=\{1..4\}}$. All the neurons in a feature map share the same set of 25 tunable weights $(w^{(1)}_{0,m}(u,v))_{(u,v)\in\{-2..2\}}$ constituting a 5×5 trainable kernel and a bias $b^{(1,m)}$. The values of the m^{th} feature map $c^{(1,m)}$ are obtained by convolving the input map $e^{(0)}$ with the respective kernel $w^{(1)}_{0,m}$ and applying an activation function ϕ to the result:

$$c^{(1,m)}(x,y) = \phi^{(1)} \left(\sum_{u=-2}^{u=2} \sum_{v=-2}^{v=2} \left(e^{(0)}(x+u+\frac{M_1}{2}, y+v+\frac{M_1}{2}) \times w^{(1)}_{0,m}(u,v) \right) + b^{(1,m)} \right) \tag{3.3}$$

This layer is therefore called a convolution layer and due to border effects its feature maps are smaller than the input map, their dimension is 28×28.

Each convolution map is followed by a subsampling map which performs a kind of averaging and reduces the dimension of the respective convolution map by a factor two. This subsampling is relevant since it reduces the computational cost and the sensitivity to affine transformations. Moreover, the exact position of the extracted feature is not as important as its relative position. Hence, the subsampling layer contains maps of size 14×14 and each map m has only one weight $w^{(2)}_{m,m}$ and one bias $b^{(2,m)}$. The value of a given

unit from a given map in this layer is:

$$s^{(2,m)}(x,y) = \phi^{(2)}\Big(\Big(w^{(2)}_{m,m} \sum_{u=0}^{u=1} \sum_{v=0}^{v=1} c^{(1,m)}(2x+u, 2y+v) \Big) + b^{(2,m)} \Big) \tag{3.4}$$

Layers C_3 and S_4 are implemented similarly to layers C_1 and S_2 respectively. Each of this layers contain 16 convolution and subsampling maps of dimensions 10×10 and 5×5 respectively. The connection strategy between layers S_2 and C_3 was chosen manually. According to the authors a partial connection scheme reduces the number of connections in the network and most importantly it forces a break of symmetry in the network. Furthermore, it allows feature maps in C_3 to extract different combination of features from different sets of feature maps in the previous layer.

The convolution layer C_5 has 120 feature maps that are fully connected to the S_4 layer. Each map m in layer C_5 has 16 weight matrices of size 5×5, each weight matrix is used to connect to a particular feature map m' from layer S_4. The last hidden layer F_6 has 84 neurons and has full connection with layer C_5.

The reason behind the choice of 84 neurons at this stage is that the desired pattern class is a bitmap of size 7×12. Finally in, the output layer, there are ten Euclidean radial basis function (RBF) units which correspond to the ten numerical digits.

In total, this architecture has $345,308$ connections, but thanks to the weight sharing mechanism, there are only $60,000$ free parameters.

3.4 Training overview

Supervised versus unsupervised learning

There are two training approaches for neural networks: supervised and unsupervised training. In supervised training, the network has access to the inputs as well as to the target outputs. It processes the inputs and compares its outputs to the target outputs. Error generated from the network is then used to update its parameters. This process is iterated until the disparity between the actual outputs and the target outputs is reduced to a predefined value. In unsupervised training, the network has access only to the inputs. The purpose of such training is either to discover underlying structure of the data, or to encode the data, or to compress it, or to transform it. A famous class of networks based on unsupervised learning is the Kohonen Self Organizing Maps (SOMs).

Supervised training algorithms are the most used, especially in pattern recognition, where the input patterns are usually labeled into classes. For this reason, we focus in this work on the supervised training methodology.

Backpropagation

Hence, in supervised learning, it is easy to compute the error of the output layer since the target outputs are known. The question then is what the target values of the hidden units are. This was an unsolved question for thirty years until the elaboration of the backpropagation algorithm in the eighties. In this algorithm, the error value for a given node i in a hidden layer is simply the weighted sum of the errors of all nodes connected from node i, which means, all nodes that have an incoming connection from node i.

There are nowadays, a large variety of training algorithms such as Quasi-Newton, Delta Bar Delta, Quick Propagation, Levenberg-Marquardt, the backpropagation algorithm is still the best known and most used training algorithm.

Error function and Gradient descent

Before applying backpropagation, an error function as well as an optimization algorithm must be chosen. In fact, learning in neural networks can be formulated as a global optimization of an error function over a high-dimensional space of weights.

The most used error function in CNNs is the mean square error (MSE) defined as the average of the square of the error. Let's consider a training set of N_{tr} elements and a classification network with N_{Class} classes. Let's note $O_{h,k}$, the output value of the unit h of the network processing a pattern k and $D_{h,k}$ its target value. The MSE is then:

$$MSE = \frac{1}{N_{tr} \times N_{Class}} \sum_{k=1}^{N_{tr}} \sum_{h=1}^{N_{Class}} (O_{h,k} - D_{h,k})^2 \tag{3.5}$$

The most used optimization algorithm is the gradient descent. It is based on the observation that if a real-valued function $F(\mathbf{x})$ is defined and differentiable in a neighborhood of a point $\mathbf{a}$, then $F(\mathbf{x})$ decreases fastest if one goes from $\mathbf{a}$ in the direction of the negative gradient of F. A sequence of such moves will eventually find a minimum of some sort. There are two kinds of gradient descent:

- Batch gradient descent which requires to sweep through the whole training set in order to evaluate the true gradient defined as the sum of

the gradients caused by each individual training example. The update of parameters is function of the true gradient.

- Stochastic gradient descent which takes randomly a training example and estimates the true gradient based on the error of that example, and then the weights are updated.

In the stochastic mode, the estimation of the gradient is noisy, the weights may not move precisely down the gradient at each iteration. Contradictorily, this may be advantageous. In fact, the goal of training is to locate one of the minima of the error function. Batch learning will discover the minimum of whatever basin where the weights are initially placed. In stochastic learning, the noise present in the updates can result in the weights jumping into the basin of another, possibly deeper, local minimum. This has been demonstrated in certain simplified cases [HK93], [CBD+90]. Moreover, the stochastic mode, also called online mode, is faster than the batch mode. For these two reasons, the stochastic mode is most used and it has been chosen for training the systems proposed in this work.

Learning rate and momentum

The weights are then updated by a step in the opposite direction of the gradient of the error function. The difficult part is to decide how large the steps should be. Large steps may converge more quickly, but may also overstep the solution or if the error surface is very eccentric, go off in the wrong direction. A classic example of this in neural network training is where the algorithm progresses very slowly along a steep, narrow, valley, bouncing from one side across to the other. In contrast, very small steps may go in the correct direction, but they also require a large number of iterations. In practice, the step size is proportional to a special constant: the learning rate. The correct setting for the learning rate is application-dependent, and is typically chosen by experiment; it may also be time-varying, getting smaller as the algorithm progresses.

The algorithm is also usually modified by inclusion of a momentum term: this encourages movement in a fixed direction, so that if several steps are taken in the same direction, the algorithm "picks up speed", which gives it the ability to sometimes escape local minimum, and also to move rapidly over flat spots and plateaus.

So more explicitly, the update of any matrix of weights $w^{(l)}_{m',m}$ connecting the map m' from the layer l to the map m from the layer $l-1$ is:

$$w^{(l)}_{m',m} \leftarrow w^{(l)}_{m',m} + \Delta w^{(l)}_{m',m} = w^{(l)}_{m',m} - \eta \frac{\partial E_p}{\partial w^{(l)}_{m',m}} + \alpha (\Delta w^{(l)}_{m',m})_{previous} \quad (3.6)$$

where η is the learning rate, E_p is the mean square error of a given pattern p and α is the momentum rate.

The training of CNNs is very similar to the training of other types of NNs, such as ordinary MLPs. The only difference to take into account within the LeCun model is the weight sharing in the convolution and subsampling layers. LeCun *et al.* proposed in [LBOM98] some justified tricks to perform an efficient backpropagation, such as: shuffling the examples of the training set so that successive training examples rarely belong to the same class, normalizing the inputs, choosing the sigmoid activation function $f(x) = 1.7159 \times \tanh\left(\frac{2}{3}x\right)$.

Given that we propose several new specific architectures of CNNs to tackle our problem of text recognition, the equations governing the backpropagation algorithm will be presented in the chapters describing each architecture.

3.5 Some Applications

Since the publication of convolutional neural networks models, this kind of architecture has been used for a wide range of real-world problems. Some of them have produced state of the art performance where traditional algorithm failed or performed poorly. Here after are detailed some of these applications.

3.5.1 Handwritten Character Recognition

Fukushima was the first to tackle this problem with CNNs, by developing the neocognitron and applying it to the recognition of handwritten numerals from zero to nine. Years later, LeCun *et al.* developed a series of CNNs architectures for the recognition of handwritten Zip codes given by the US postal service in Washington DC [CBD+89], [LJB+95]. Thus, the US National Institute of Standards and Technology (NIST) provided two databases of handwritten characters for training and evaluating any handwritten character recognition systems. Afterwards, LeCun *et al.* reorganized these databases in order to enhance its distributions and come up with the Modified NIST (MNIST) database. Currently, this database is very used by the community of handwritten text recognition. The most known work in this field is the LeNet-5 [LBBH98], which was developed commercially in several banks to perform check recognition. Based on the MNIST, LeNet-5 achieves an error rate of 0.8%.

Calderón *et al.* [CRV03] developed a novel type of convolutional networks using Gabor filters as feature extractors at the first layer. They propose a

topology of six layers. The input layer, made up of 28×28 sensory nodes, receives the images of different characters that have been normalized in size. The first layer is represented by Gabor filters connected to the input layer. It is similar to a convolutional layer, except that it has no training parameters. This layer is composed by 12 sub-layers, which are in particular the Gabor filter responses to 2 different frequencies and 6 orientations, with $\sigma = 2.4$. The selection of these parameters was found to be experimentally efficient. The size of each sub-layer is 28×28. The four following layers are alternatively subsampling and convolution layers as in the LeNet-5. The output layer is also composed of 84 neurons, representing the output of the net as a gray-scale image. Moreover, they used the boosting algorithm. Boosting is a class of committee machines, which combine decisions of different experts to come up with a superior overall decision, distributing the learning task between experts. Drucker et al. [DSS93] developed a specific boosting method for the LeNet-4 architecture of Convolutional networks, reaching better results. Calderón *et al.* use a type of boosting by filtering method then the output of the boosting machine was obtained by simply adding the results of the three experts. They report a recognition error rate of 0.68%, which is better than the LeNet-5. Simard *et al.* [SSP03] further reduced the error recognition error rate on the MNIST database to 0.4%. They propose a simpler topology with only four layers. The two first layers are convolutional layers then the two last layers represent a fully connected MLP with 100 hidden units and ten output neurons. In addition, they use the cross-entropy function instead of the mean square error as an error function.

3.5.2 Face Detection and Recognition

Garcia and Delakis [GD04] presented a face detection method using Convolutional Neural Networks with face and non-face examples and a specific bootstrapping algorithm. Being particularly robust with respect to noise and partial occlusions and producing very few false alarms, this technique, so far, shows the best performance on difficult public face databases. It reaches a detection rate of 90.3% with eight false alarms based on the CMU test set [RBK96]. This system dubbed Convolutional Face Finder (CFF) is essentially a CNN with six layers with an input layer of size 32×36 pixels. It is based on the CNN model of LeCun *et al.* [LBBH98] but has a different number of feature maps and neurons and notably a different connection scheme between the layers. The network has a total of 951 trainable parameters which are updated through an online error backpropagation algorithm with momentum.

Osadchy *et al.* [OCM07] also presented a face detection system based

on CNNs. They employed a network architecture similar to LeNet-5 and trained it to map face images with known head pose onto points in a low-dimensional manifold parameterized by pose and non-face image onto points far away from that manifold. Once the CNN has been trained, it can classify image sub-regions as face or non-face by applying the mapping to them and calculating the distance between the projected vector and the analytical manifold. However, the performance of the method in terms of detection rate and false alarm rate measured on the CMU data set is inferior to the system proposed by Garcia *et al.* [GD04].

In [Neu98], Neubauer evaluated the performance of CNNs for visual pattern recognition and implemented a modified version of the neocognitron, named NEO. This network has achieved a face recognition error rate of 4.5% on a constrained test set of face patterns of size 32×32, whereas for unconstrained set (with variations in pose and illumination), the recognition error rate dropped to 59.7%. When the system is restricted to the identification of one person out of 18, the performance reached 87.6% with a correct rejection rate of 97.3%.

Another well known face recognition system based on CNNs, was developed by Lawrence *et al.*. They combine local image sampling which normalizes the contrast of the input image, a self-organization map neural network (SOM) which aims at reducing the sensitivity of the system towards transformations, and a convolutional neural network which performs the face classification. With five faces per class in training and testing, they achieved a recognition rate of 96.2% when classifying 40 individuals.

3.5.3 Face Expression Analysis

Facial expression provides cues about emotion, regulates interpersonal behavior, and communicates psychopathology. In [Fas02], Fasel developed a system based on CNNs to recognize and classify human facial expression into seven groups: happiness, sadness, fear, disgust, surprise, anger and neutral. These basic emotions were postulated by Ekman and Friesen [EF71] and were considered as universal across ethnic groups and cultures. Fasel proposed a topology of five layers: two pairs of convolutional and subsampling layers and an output layer of six neurons fully connected to the feature maps of the previous layer, each corresponding to an emotion class. The input layer is a 64×64 pixels image. The network has 47787 neurons with 1902 trainable weights. Tested on the JAFFE facial expression database [LAKG98], their system shows robustness against translation, scale and rotation.

Matsugu *et al.* [MMMK03] used also CNNs for face expression analysis. Moreover, they incorporate two functionalities in the network: facial expres-

sion recognition and face detection. They use a slightly modified training methodology with fixed weights in the subsampling layers. They report a facial expression recognition rate of 97.6%, when tested on 5600 still face images.

3.5.4 Others

CNNs are also used to tackle a variety of other computer vision problems such as text detection [LL99], [DG07], logo detection in TV programs [DG06], hand detection and tracking [NP95], and even in biomedical image processing to detect masses on mammograms [SCP+96], lung nodules [LFLM92], [LFLM93], microcalcifications [CLS+95];

3.6 Conclusion

We presented in this chapter a brief introduction to artificial neural networks. These networks are able to learn and produce complex decision boundaries from input data. However, as the dimension of the inputs increases as in the field of image processing, the conventional neural networks performance decreases and the need of a more sophisticated architecture rises. Hence, a new 2D neural network architecture, named convolutional neural networks (CNNs) was developed. The CNNs are widely used in visual pattern recognition, and some of them exhibited state of the art performances. These good results encouraged us to orient this work to tackle the problem of image text recognition using CNNs.

Chapter 4

Video Text Binarization

Il y a tant de vagues et de fumée
qu'on n'arrive plus à distinguer
le blanc du noir et l'énergie du désespoir.

There is so many waves and smoke that
we no longer distinguish
the white of the black and the energy of the despair.
`[Michel Berger] (1947--1992)`

4.1 Introduction

Current optical character recognition systems (OCRs) require a binarization step that aims at separating the text pixels from the background pixels of the processed image. In fact, most OCR packages on the market work only on bi-level images. The simplest way to perform image binarization is to choose a threshold value, and classify all pixels with values above this threshold as white, and all other pixels as black. The problem then is how to select the correct threshold. Most of the text image binarization methods rely on global or local discriminating thresholds. These thresholds are determined according to some statistical analysis of the luminance or chrominance distribution generally based on histograms, without taking into account the characters shapes.

In this chapter, we propose a novel automatic binarization scheme for color text images, based on supervised learning, without making any assumptions or using tunable parameters. Our contributions to the current state of the art are the following:

- Developing a new convolutional neural network architecture that pro-

cesses color text images to directly produce binary text images.

- Using conjointly color distribution and the geometrical properties of characters.
- Insuring robustness to noise, to complex backgrounds and to luminance variations.

The remainder of this chapter is organized as follows. Section 4.2 describes in detail the architecture of the proposed neural network, especially the two new layers that we added to the classical convolutional neural network to deal with the specific problem of binarization, namely, the up-sampling and the inverse convolution layers. It explains also the implementation parameters and the training process. Experimental results are reported in Section 4.3. Conclusions are drawn in Section 4.4.

4.2 CTB: Convolutional Text Binarizer

4.2.1 Architecture of the CTB

The proposed neural architecture, called Convolutional Text Binarizer (CTB), is based on convolutional neural network architecture [LBBH98, GD04]. As shown in figure 4.1, it consists in five different heterogeneous layers. Each layer contains feature maps which are the results of either convolution, sub-sampling, up-sampling or inverse convolution operations. Applying and combining these automatically learnt operations insure the extraction of robust features, leading to the automatic construction of the binary image.

The first layer is the input layer E; it consists of $N_E = 3$ input maps, each of them corresponding to one color channel of the image, according to the chosen color space (RGB, YUV, etc.). Their pixel values are normalized to the range [-1, 1].

The second layer is a convolution layer C_1. It is composed of NC_1 maps. Each unit in each map is connected to a $M_1 \times M_1$ neighborhood (biological local receptive field) in the maps of previous layers. Furthermore, the trainable weights (convolutional mask) forming the receptive field, are forced to be equal for the entire map (weight sharing). A trainable bias is added to the results of each convolutional mask. Each map can be considered as a feature map that has a learnt fixed feature extractor that corresponds to a pure convolution with a trainable mask, applied over the maps in the previous layer. Multiple maps lead to the extraction of multiple features.

The third layer is a sub-sampling layer S_2. It is composed of NS_2 feature maps. It performs local averaging and sub-sampling operations. We use this

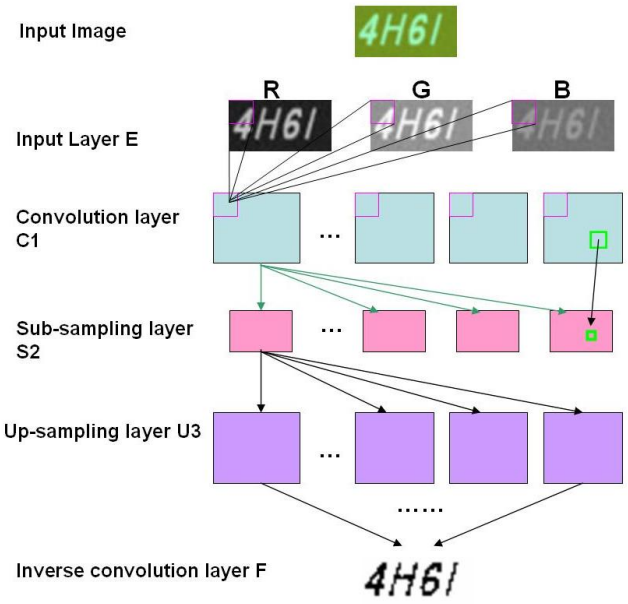


Figure 4.1: The CTB Architecture

sub-sampling layer to reduce by two the spatial resolution which reduces the sensitivity to shifts, distortions and variations in scale and rotation. It is fully connected to the previous layer C_1 which results in combining the local features extracted in the layer C_1 and extracting more complex information.

At this stage, the required features are extracted from the input image and the following steps will perform mainly the construction of the binary image.

The fourth layer is an up-sampling layer U_3. It contains NU_3 maps U_{3k}. Each set of non overlapping $M_2 \times M_2$ unit of each map is connected to a unit in each map of the previous layer (cf. Fig. 4.2). The $M_2 \times M_2$ synaptic weights and the bias are trainable and are shared over each map. In addition to the image construction role performed here, this layer aims to increase the system non linearity in order to be more robust to distortions.

The fifth and final layer is the inverse convolution layer that contains the output map F. It is fully connected to the previous layer based on an inverse scheme of the convolution. In fact, in the convolution layer, each neuron of the convolution map is connected to a set of $M_1 \times M_1$ neighbor element in the input map whereas each set of $M_1 \times M_1$ neuron of the inverse convolution map F is connected to one unit of the previous layer (cf. Fig. 4.3) in order to obtain an output map of the size of the input image. Sigmoid activation

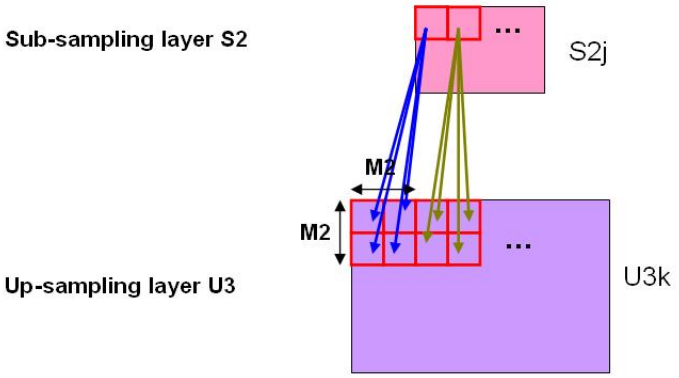


Figure 4.2: The up-sampling layer

function is used and the neuron outputs are continuous and vary between -1 and 1.

4.2.2 A Specific Implementation of the CTB

The architecture of the CTB consists of five layers which will extract the relevant features for binarization, reduce the sensitivity to different kind of distortions and build the binary image.

Layer E which contains 3 maps $(e^{(ch)})_{ch\in\{1..3\}}$ corresponding to the normalized color channels of the input image.

Layer C_1: This is the first hidden layer, it consists of $NC_1 = 30$ maps fully connected to the previous layer. Each map m have three matrices of 25 weights $(w^{(1)}_{ch,m}(u,v))_{ch\in\{1..3\},(u,v)\in\{-2..2\}}$, constituting three $M_1 \times M_1 = 5 \times 5$ trainable kernels and corresponding to the convolutional masks operating on the three maps in the previous layer.
Thus, the values $c^{(1,m)}(x,y)$ of a given feature map $c^{(1,m)}$ are obtained by convolving the input maps $e^{(ch)}_{ch\in\{1..3\}}$ with their respective kernels $(w^{(1)}_{ch,m})_{ch\in\{1..3\}}$, adding a bias $b^{(1,m)}$ and applying an activation function Φ as shown in the following formula:

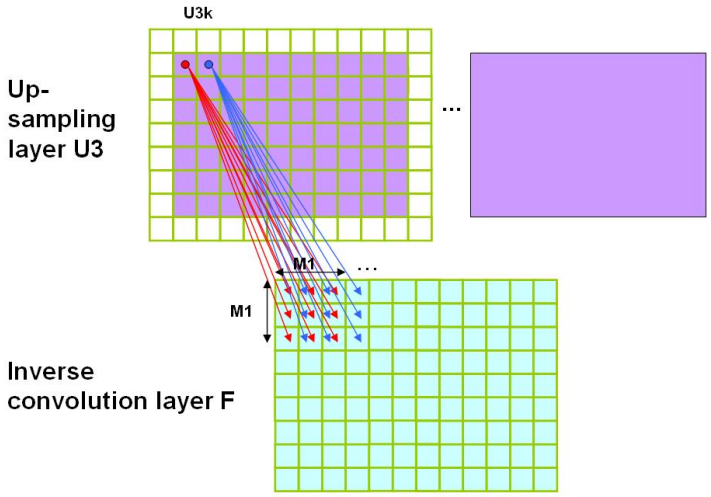


Figure 4.3: The inverse convolution layer

$$c^{(1,m)}(x,y) = \Phi\left(\sum_{ch=1}^{3}\sum_{u=-2,v=-2}^{u=2,v=2}\left(e^{(ch)}(x+u+\frac{M_1}{2}, y+v+\frac{M_1}{2}) \times w^{(1)}_{ch,m}(u,v)\right) + b^{(1,m)}\right) \tag{4.1}$$

The activation function here is the identity function because we don't need to add non linearity at this stage.

Layer S_2: This is the second hidden layer, it consists of $NS_2 = 30$ maps fully connected to the previous layer. Each map m in this layer has $NC_1 = 30$ weights $(w^{(2)}_{m',m})(0)$, constituting the trainable coefficients of the sub-sampling mask operating on the maps of the previous layer.
The values $s^{(2,m)}(x,y)$ of the feature map $s^{(2,m)}$ are obtained by performing a weighted sub-sampling of factor two, $M_2 \times M_2 = 2 \times 2$, adding a bias $b^{(2,m)}$ and applying an activation function Φ as described in the following formula:

$$s^{(2,m)}(x,y) = \Phi\left(\sum_{m'=1}^{NC_1}\left(w^{(2)}_{m',m}(0)\sum_{u=0,v=0}^{u=1,v=1} c^{(1,m')}(2x+u, 2y+v)\right) + b^{(2,m)}\right) \tag{4.2}$$

The activation function here is the sigmoid function presented in chapter 3.

Layer U_3: This is the third hidden layer, it consists of $NU_3 = 30$ maps fully connected to the previous layer. Each map m in this layer have $NS_2 = 30$ matrices of 4 weights $(w^{(3)}_{m',m})(u,v)_{m' \in \{0..NS_2\},(u,v)\in\{0..1\}}$, constituting a $M_2 \times M_2 = 2 \times 2$ trainable kernel and corresponding to the up-sampling mask operating on the previous layer.
The values $u^{(3,m)}(x,y)$ of the feature map $u^{(3,m)}$ are obtained by performing a weighted up-sampling, adding a bias $b^{3,m}$ and applying an activation function Φ as shown in the following formula:

$$\begin{aligned} &for\ u \in \{0,1\}\ and\ v \in \{0,1\}, \\ &u^{(3,m)}(2x+u, 2y+v) = \Phi \left(\sum_{m'=1}^{NS_2} \left(s^{(2,m')}(x,y) \times w^{(3)}_{m',m}(u,v) \right) \right. \\ &\qquad \left. + b^{(3,m)} \right) \end{aligned} \tag{4.3}$$

The activation function here is the sigmoid function presented in chapter 3.

Layer F_4: This is the output layer, it consists of only one map fully connected to the previous layer. The unique map 0 in this layer have $NU_3 = 30$ matrices of 25 weights $((w^{(4)}_{m',0})(u,v))_{m' \in \{0..NU_3\},(u,v)\in\{-2..2\}}$, constituting a $M_1 \times M_1 = 5 \times 5$ trainable kernel and corresponding to the inverse convolution mask operating on each map of the previous layer.
The values $f^{(4,0)}(x,y)$ of the output map $f^{(4,0)}$ are obtained by performing an inverse convolution of the previous layer maps $u^{(3,m)}_{m\in\{1..30\}}$ with their respective kernels $(w^{(3)}_{m',m})_{m\in\{1..30\}}$, adding a bias $b^{(4)}$ and applying an activation function Φ which is the sigmoid here, as shown in the following formula:

$$\begin{aligned} f^{(4,0)}(x,y) = \ \Phi & \left(\sum_{m=1}^{NU_3} \sum_{u=-2,v=-2}^{u=2,v=2} \left(u^{(3,m')}(x-u-\frac{M_1}{2}, y-v-\frac{M_1}{2}) \right. \right. \\ & \left. \left. \times w^{(4)}_{m',0}(u,v) \right) + b^{(4)} \right) \end{aligned} \tag{4.4}$$

4.2.3 Training Phase

Data

The training phase is supervised; therefore it is required that the training set contains pairs of images: the color text images which represents the inputs and their corresponding binary images which represents the targets. To avoid hard and tremendous annotation task and manual text binarization,

and because we are convinced of the good generalization ability of the neural network, we choose to use synthetic images that we build automatically. Thus, a set of binary text images of size $W \times H = 48 \times 24$ is constructed using different fonts. For each one, we randomly choose a text color and a background color and we apply different types of noise (uniform and Gaussian noises), and then we apply smoothing filters in order to obtain images looking like real ones and presenting a lot of variability. Figure 4.4 shows some examples of training images. Our training set contains 5000 text color images divided in 4500 images for the enrolling and 500 images for the validation. In fact, to avoid overfitting the training data and to increase the generalization ability of the system, a random part of the whole training set is kept as validation set. This validation set is only used to evaluate the network performance after each iteration. Its images are not learnt, which means that no weights are updated as function of the error related to the validation set images.

Figure 4.4: Some examples from the training set

Training

The training phase was performed using the classical backpropagation algorithm which means that the errors propagate backwards from the output neurons to the inner units. The update of the trainable weights is computed as a function of the gradient's error of the network with respect to

these weights according to the gradient descent algorithm which aims to find weights that minimize the error. We choose to apply the online backpropagation which correspond to the stochastic gradient descent algorithm. The major point in this algorithm is that the weights update is performed after each example. Furthermore, we apply the momentum modified as described in [LBBH98] to improve the convergence speed.

The objective of the network being to obtain final values $f^{(4,0)}(x,y)$ equal to the desired values $d(x,y)$, we classically choose to minimize the Mean Square Error (MSE) between obtained and desired values over a validation set of $N_v = 500$ images.

We define the error E_p of a training image p as:

$$E_p = \frac{1}{2} \sum_{(x,y)=(0,0)}^{(W,H)} (f^{(4,0)}(x,y)_p - d(x,y)_p)^2 \tag{4.5}$$

Then, the MSE over a given validation set is:

$$MSE = \frac{1}{N_v} \sum_{p=1}^{N_v} E_p \tag{4.6}$$

At each iteration, the three color channels of a given text image are presented to the network as inputs and its corresponding binary image as the desired output map. The weights update is then an iterative procedure, where their values are modified by a small variation in the opposite direction of the steepest gradient ∇E of the error E. More explicitly, the update of any matrix of weights $w^{(l)}_{m',m}$ connecting the map m' from the layer l to the map m from the layer $l-1$ is:

$$w^{(l)}_{m',m} \leftarrow w^{(l)}_{m',m} + \Delta w^{(l)}_{m',m} = w^{(l)}_{m',m} - \eta \frac{\partial E_p}{\partial w^{(l)}_{m',m}} + \alpha (\Delta w^{(l)}_{m',m})_{previous} \tag{4.7}$$

where η is the learning rate, E_p is the mean square error of a given pattern p and α is the momentum rate.

We detail below the formulas governing the backpropagation of the CTB.

The partial derivative of a pattern error E_p according to a given weight $w^{(4)}_{m',0}(u,v)$ connecting the map m' in layer $U^{(3)}$ to the unique map 0 of layer F_4 at the kernel position (u,v) is:

$$\frac{\partial E_p}{\partial w^{(4)}_{m',0}(u,v)} = \sum_{(x,y)=(0,0)}^{(W,H)} [(f^{(4,0)}(x,y) - d(x,y))(\frac{\partial f^{(4,0)}(x,y)}{\partial w^{(4)}_{m',0}(u,v)})] \tag{4.8}$$

Where,

$$\frac{\partial f^{(4,0)}(x,y)}{\partial w^{(4)}_{m',0}(u,v)} = u^{3,m'}(x-u-\frac{M_1}{2}, y-v-\frac{M_1}{2}) \times \phi'(A^{(4,0)}(x,y)) \quad (4.9)$$

where, ϕ' is the derivative of the activation function used in the last layer, and $A^{(4,0)}(x,y)$ is the activation at the unit (x,y) of the map 0 of the layer $F^{(4)}$. For seek of simplicity, we have omitted the index p from the formulas, however, they are still relative to a given pattern p.

We define $e^{(4,0)}(x,y) = (f^{(4,0)}(x,y) - d(x,y))$ which is the local error, and $\delta^{(4,0)}(x,y) = e^{(4,0)}(x,y) \times \phi'(A^{(4,0)}(x,y))$, which corresponds to the local gradient.
So,

$$\begin{aligned}\Delta w^{(4)}_{m',0}(u,v) &= -\eta \textstyle\sum_{x,y} \left(\delta^{(4,0)}(x,y) u^{(3,m')}(x-u-\tfrac{M_1}{2}, y-v-\tfrac{M_1}{2})\right) \\ &\quad +\alpha(\Delta w^{(4)}_{m',0}(u,v))_{previous} \end{aligned} \quad (4.10)$$

In a similar way, we compute the derivative of the error according to the bias. The update is then the following:

$$\Delta b^{(4,0)} = -\eta \sum_{x,y} (\delta^{(4,0)}(x,y)) + \alpha(\Delta b^{(4,0)})_{previous} \quad (4.11)$$

Once the weights matrices and the bias of the unique map of the last layer have been updated, the next step is to backpropagate the error to the layer $U^{(3)}$.
The local error at position (x,y) for a given map m is shown in the following equation :

$$e^{(3,m)}(x,y) = \sum_{(u,v)=(-2,-2)}^{(2,2)} (\delta^{(4,0)}(x+u+\frac{M_1}{2}, y+v+\frac{M_1}{2}) \times w^{(4)}_{m,0}(u,v)) \quad (4.12)$$

The local gradient is computed by multiplying the local error by the derivative of the activation function:

$$\begin{aligned}\delta^{(3,m)}(x,y) = \phi'(A^{(3,m)}(x,y)) \times \sum_{(u,v)=(-2,-2)}^{(2,2)} \Big(& \delta^{(4,0)}(x+u+\tfrac{M_1}{2}, y+v+\tfrac{M_1}{2}) \\ & \times w^{(4)}_{m,0}(u,v)\Big) \end{aligned} \quad (4.13)$$

Then, the weights update is:

$$\begin{aligned}\Delta w^{(3)}_{m',m}(u,v) &= -\eta \textstyle\sum_{x,y} \left(\delta^{(3,m)}(x,y) s^{(2,m')}(x/2-u, y/2-v)\right) \\ &\quad +\alpha \Delta w^{(3)}_{m',m}(u,v))_{previous} \end{aligned} \quad (4.14)$$

In a similar way, we compute the update of the bias:

$$\Delta b^{(3,m)} = -\eta \sum_{x,y} (\delta^{(3,m)}(x,y)) + \alpha(\Delta b^{(3,m)})_{previous} \tag{4.15}$$

Now, we have to backpropagate the error to the sub-sampling layer S_2. The local error at position (x, y) for a given map m from the layer S_2 is shown in the following equation :

$$e^{(2,m)}(x,y) = \sum_{m''=0}^{NU_3-1} \sum_{(u,v)=(0,0)}^{(1,1)} (\delta^{(3,m'')}(2x+u, 2y+v) \times w^{(3)}_{m,m''}(u,v)) \tag{4.16}$$

The local gradient is computed by multiplying the local error by the derivative of the activation function:

$$\delta^{(2,m)}(x,y) = \phi'(A^{(2,m)}(x,y)) \sum_{m''=0}^{NU_3-1} \sum_{(u,v)} (\delta^{(3,m'')}(2x+u, 2y+v) \times w^{(3)}_{m,m''}(u,v)) \tag{4.17}$$

Then, the update of the weight $w^{(2)}_{m',m}$ connecting each four element from the map m' from layer C_1 to one element from the map m from layer $S_2$is:

$$\begin{aligned} \Delta w^{(2)}_{m',m} &= -\eta \textstyle\sum_{x,y} \Big(\delta^{(2,m)}(x,y) \sum_{(p,q)\in\{0,1\}} c^{(1,m')}(2x+p, 2y+q)\Big) \\ &\quad + \alpha \Delta w^{(2)}_{m',m})_{previous} \end{aligned} \tag{4.18}$$

In a similar way, we compute the update of the bias:

$$\Delta b^{(2,m)} = -\eta \sum_{x,y} (\delta^{(2,m)}(x,y)) + \alpha(\Delta b^{(2,m)})_{previous} \tag{4.19}$$

Finally, we backpropagate the error to the first hidden layer C_1. The local error at position (x, y) for a given map m from the layer C_1 is shown in the following equation :

$$e^{(1,m)}(x,y) = \sum_{m''=0}^{NS_2-1} \delta^{(2,m'')}(x/2, y/2) \times w^{(2)}_{m,m''} \tag{4.20}$$

The local gradient is computed by multiplying the local error by the derivative of the activation function:

$$\delta^{(1,m)}(x,y) = \phi'(A^{(1,m)}(x,y)) \sum_{m'=0}^{NS_2-1} \delta^{(2,m')}(x/2, y/2) \times w^{(2)}_{m',m} \tag{4.21}$$

Then, the update of the weight matrices $w^{(1)}_{ch,m}(u,v)$ where $(u,v) \in \{-2..2\}$ connecting the input map ch from to the map m from layer $C_1$is:

$$\begin{aligned}\Delta w^{(1)}_{ch,m}(u,v) &= -\eta \textstyle\sum_{(x,y)} \big(\delta^{(1,m)}(x,y) e^{(ch)}(x+u+\tfrac{M_1}{2}, y+v+\tfrac{M_1}{2})\big) \\ &\quad +\alpha \Delta w^{(1)}_{ch,m}(u,v))_{previous} \end{aligned} \tag{4.22}$$

In a similar way, we compute the update of the bias:

$$\Delta b^{(1,m)} = -\eta \sum_{x,y} (\delta^{(1,m)}(x,y)) + \alpha(\Delta b^{(1,m)})_{previous} \tag{4.23}$$

The figure 4.5 illustrates the relation of some parameters governing the backpropagation algorithm for the CTB.

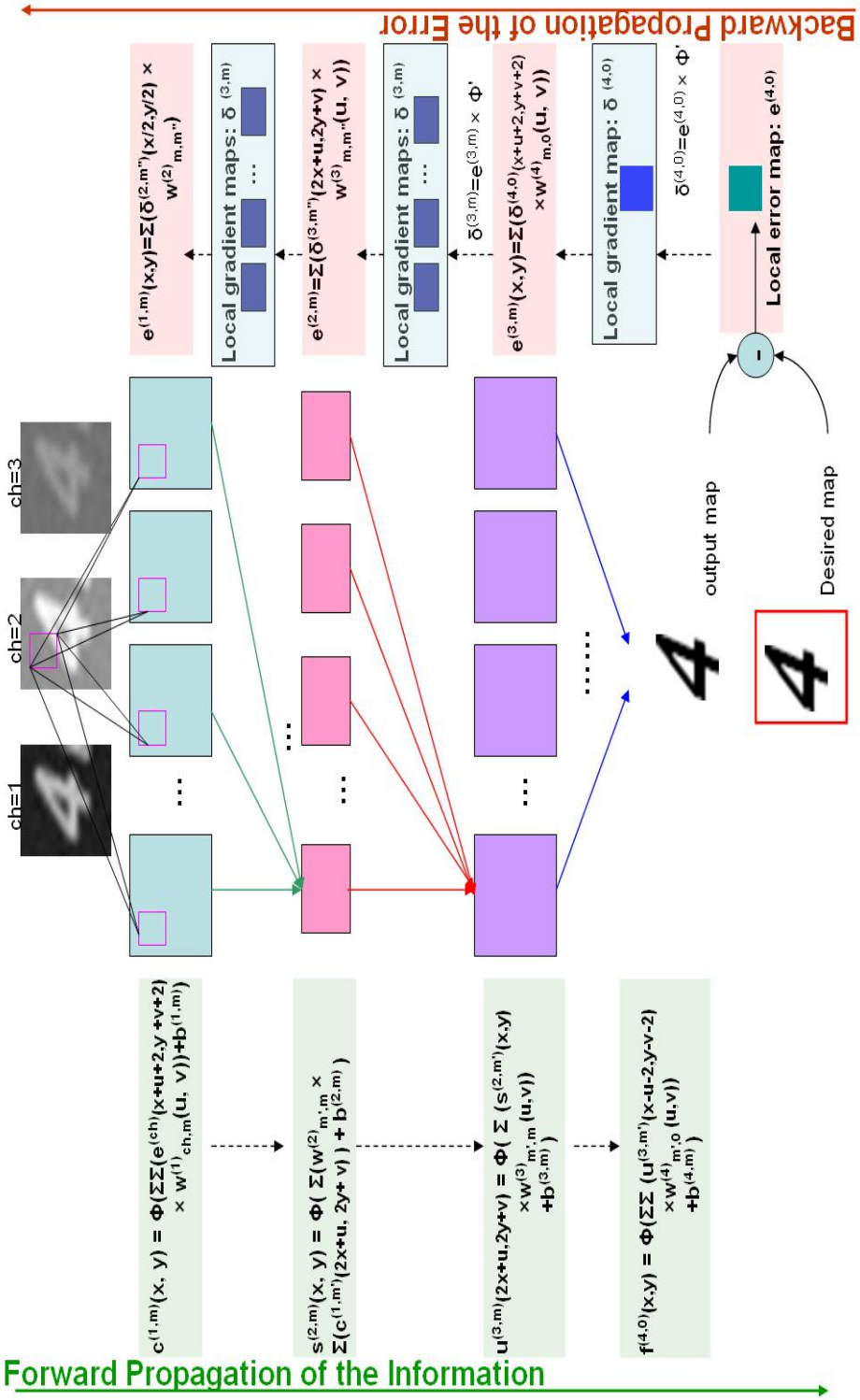


Figure 4.5: Error back-propagation in the CTB

Convolution windowsize	3×3	5×5	7×7	9×9
MSE on Training set	0.058	0.030	0.035	0.046

Table 4.1: Performance as function of the convolution windowsize

4.2.4 The choice of parameters

As previously detailed, we choose to build $NC_1 = NS_2 = NU_3 = 30$ maps in layers C_1, S_2 and U_3. We use linear activation functions in C_1 and the sigmoid in S_2, U_3, and F. The convolution and the inverse convolution window size $M_1 \times M_1$ is 5×5. The sub-sampling and the up-sampling factor is two in each direction.

The different parameters governing the proposed architecture, i.e., the size of the receptive fields, the number of maps, as well as the learning rate, have been experimentally chosen. We expose below some of the most relevant experiments which justify our choices. All these experiments have been running over 100 iterations and for each one, we vary one parameter and we keep the best values for the remaining parameters.

Let us focus first on the convolution window size. On one hand, as this size becomes larger, the robustness against distortion is enhanced, but more image details are lost. On the other hand, as this size becomes smaller, more precision is gained at the cost of less robustness against distortion. Furthermore, this size depends also on the characters size because the aim of the system is to binarize text images and should therefore extract features related to the characters of the processed image. Provided that the characters sizes in the training set ranges approximately from (4×10) to (14×16), the convolution window sizes considered here are: $(3 \times 3), (5 \times 5), (7 \times 7)$ and (9×9).

Table 4.1 shows the performance of the system tested on the training set as function of the size of the convolution window. This performance is evaluated through the mean square error between the image obtained and the image desired. We notice that the best result is given by the size (5×5).

When it comes to the number of maps in each layer, the stake is twofold. On one hand there is the performance and on the other hand, the computation cost. In addition, for a large number of maps per layer, the system may overlearn and the performance decreases.

Table 4.2 shows the performance of the system tested on the training set as function of the number of maps in the layers C_1, S_2 and U_3. This performance is evaluated through the mean square error between the image obtained and the image desired. We notice that the best result is given when the number of maps is 30.

Number of maps	20	25	30	35
MSE on Training set	0.058	0.034	0.030	0.038

Table 4.2: Performance as function of the number of maps per layer

Learning rate	10^{-3}	10^{-4}	10^{-5}	10^{-6}
MSE on Training set	1.117	0.040	0.030	0.078

Table 4.3: Performance as function of learning rate

Finally, we propose to study the importance of the learning rate. If the learning rate is too low, the network learns very slowly. If the learning rate is too high, the weights may keep oscillating around the error surface and the algorithm diverges. Table 4.3 shows the performance of the system tested on the training set as function of the learning rate. This performance is evaluated through the mean square error between the image obtained and the image desired. We notice that the best result is given when the number of maps is 10^{-5}.

4.2.5 Testing Phase

After training, the system is able to produce a binary image for a given color input text image as follow:

1. Resize the text image to the retina height, while preserving the aspect ratio (the size of the training input images are called retina size);
2. Feed the network with the resized text image as input;
3. Retrieve the output map of the network;
4. The last step consists in converting this output map F on a binary image according to the sign of the neurons outputs. Let $f_{i,j}$ be the value of the element (i, j) in the output map F. The pixel $P_{i,j}$ at the position (i, j) in the binary image is assigned to 0 if $f_{i,j} < 0$ and to 1 if $f_{i,j} \geq 0$.

We have to mention here that the width of the test image to process must not be equal to the retina width, because of the sharing weights property of this network. In fact, these shared matrices of weights can be applied as much as necessary according to the width of the image processed.

4.3 Experimental results

To test the performance of our method, we use two databases:

- The first one is composed of synthetic images, which allow us to evaluate the robustness of our method according to noise and contrast variations.
- The second one contains real images collected from video frames and web pages.

When testing real images, we first crop manually the text box. Then, we resize it so that its height is equal to the retina height ($H = 24$), while keeping the same aspect ratio.

We compare our method to the following well-known binarization methods: the Otsu method [Ots79], the Niblack method [Nib86], the Sauvola method [SSHP97], and the Lienhart method [LW02]. The evaluation is based on two criteria:

- the MSE between obtained binary images and desired binary images for the tests on contrast and noise
- the recognition rates of a free OCR named TESSERACT [Hp], and a trial version of a commercial one named ABBy FineReader 8.0 [Abb] for test on real text images.

4.3.1 Robustness versus noise

To test the impact of noise on the different binarization methods, we build a database of 1000 images divided into five categories. Each category contains 200 text images to which we have added an increasing Gaussian noise, using the variances 3, 5, 7, 9 and 11.

Figure 4.6 shows the stability of our method with respect to noise addition and that it gets the lowest MSE. We notice that Otsu method, which is a global thresholding technique, is the most sensitive to noise addition with a variation of 0.46 units in MSE.

4.3.2 Robustness versus contrast variation

To test the influence of contrast variations, a database of 1000 synthetic text images is used. Then the images are classified according to their contrast. The contrast is computed based on the following formula:

$$Contrast = \frac{|TxtColor - BckColor|}{TxtColor + BckColor}$$

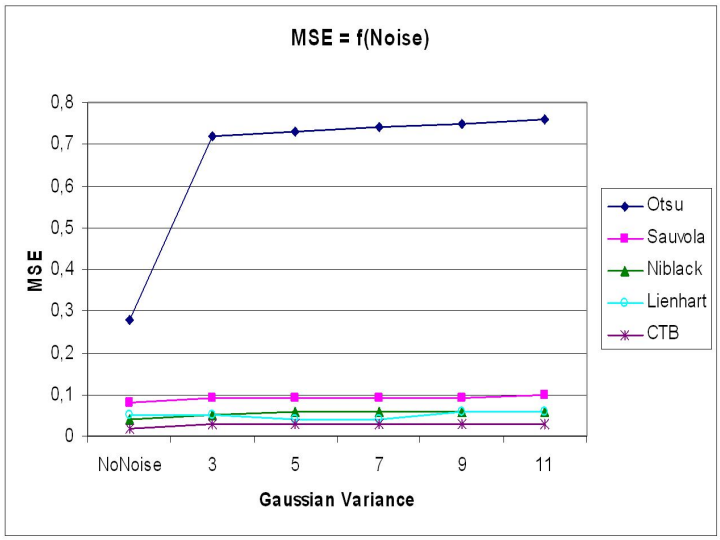


Figure 4.6: MSE versus additive Gaussian noise

where $TxtColor$ and $BckColor$ are the color of the text and the background, respectively.

Figure 4.7 shows that our method is extremely robust to contrast variation, followed by the Lienhart and the Niblack methods. Notice that generally, methods are more influenced by contrast variation than by noise variation. In fact, if we compute the relative variation rate for each curve, we find that the average relative variation rate for the noise variation curve is 3.19%, while for the contrast variation curve, it is 7.07%.

4.3.3 Recognition efficiency

It is important to evaluate the recognition performance on text images that have been binarized by the CTB, since this is the major and final goal of the application. Therefore, we build a database composed of 161 images collected from videos frames and web pages, containing 2252 characters. The images contain complex background, with high chrominance variability. Some examples of this database are shown in Fig 4.8 Figure 4.9 reports the recognition rates of the two OCRs (Tesseract and Abby FineReader 8.0) applied on binarized real text images. The binarization is achieved by the CTB and the four classical binarization methods previously mentioned. Some examples of this database with their corresponding binarized images obtained by the CTB and by the Lienhart method (as it gets the second best results) are

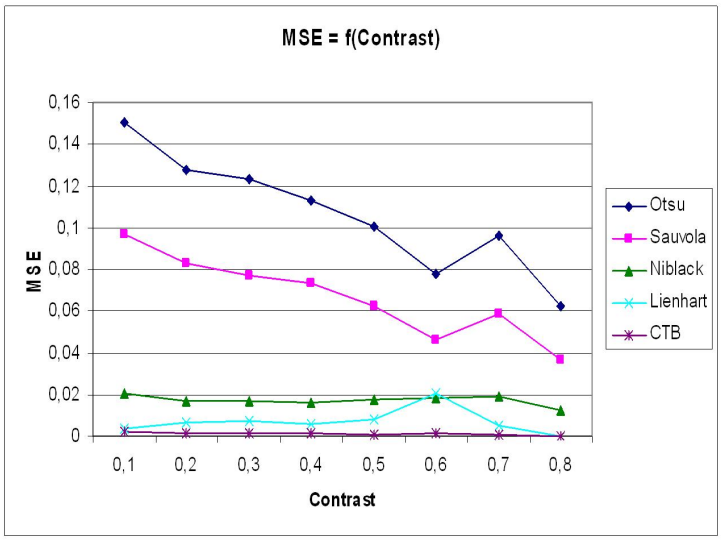


Figure 4.7: Recognition rates versus contrast variations

shown in Figure 4.10. This figure 4.9 illustrates the impact of the robust binarization provided by our method both on free and commercial OCR. We reach a recognition rate of 86.234% with the commercial OCR, while the other methods get recognition rates lower than 50%. It can be noticed that a slight gain in MSE have a tremendous impact on text recognition results. In fact, some erroneous pixels around a character, do not have high impact on the MSE, but can lead the recognizer to misclassify the character.

4.4 Conclusion

In this chapter, we have proposed a novel approach based on specific convolution neural network architecture designed for complex color text image binarization. Based on supervised learning, our system does not need any tunable parameter and takes into account both color distributions and the geometrical properties of characters.
Experimental results on noise and contrast variations show that the proposed method is robust and greatly enhance text recognition rates using classical document OCRs, compared to the state of the art binarization method. At this stage, we will keep the binarization system aside and we will focus on the character segmentation system which combined to a character recognition system can be used instead of the binarization system combined with document OCR as discussed in the next chapters.

Figure 4.8: Some examples from the collected database for text recognition

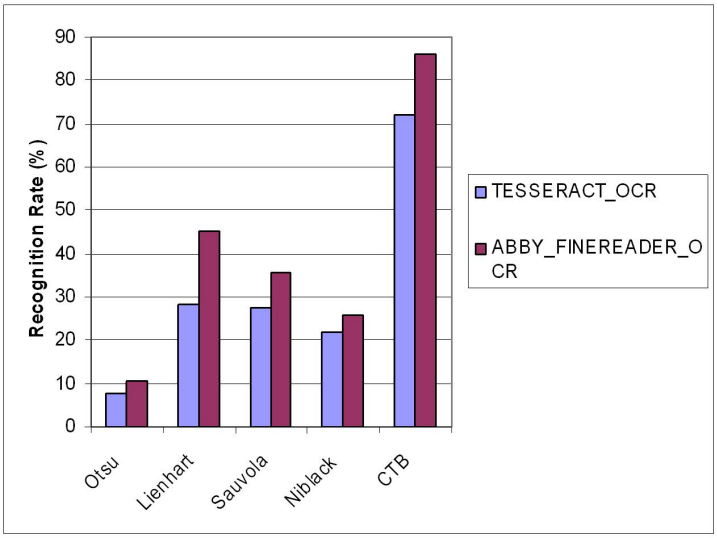


Figure 4.9: Recognition rates on binary images

Figure 4.10: Some examples of collected text images with their binary corresponding images obtained with the CTB.

Chapter 5

Video Text Segmentation

Tout devrait être fait aussi simple que possible,
mais pas plus simple.

Everything should be made as simple as possible,
but not one bit simpler.
[Albert Einstein] (1879--1955)

5.1 Introduction

When the recognition is based on characters, a preprocessing step called segmentation is usually required. This step consists on splitting the text image into isolated character images. This task is challenging because of two reasons:

- on one hand text images have usually complex background, noise, and illumination variation;
- on the other hand a part of a character image may be confusing with another character. For example, a part of the letter 'm' could be 'n', and a part of the letter 'w' could be 'v'. In a similar way, the neighborhood of some characters may be confusing with another one, such as 'ri' which can be confused with 'n'.

In this chapter, we propose an automatic segmentation system for characters in text color images cropped from natural images or videos based on a new neural architecture insuring fast processing and robustness against noise, variations in illumination, complex background and low resolution.

The remainder of this chapter is organized as follows. Section 5.2 describes in detail the architecture of the proposed neural network and the

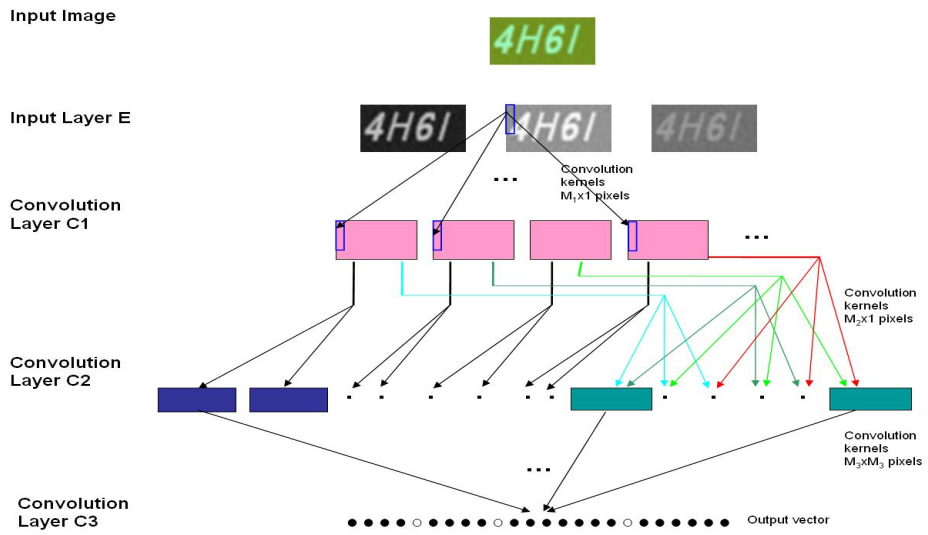


Figure 5.1: Architecture of the segmentation neural network

training process. It explains also how to use this system combined with a character recognition system. Experimental results are reported in Section 5.3. Conclusions are drawn in Section 5.4.

5.2 CTS: Convolutional Text Segmenter

5.2.1 Architecture of the CTS

Our character segmentation approach relies on a novel architecture of convolutional neural networks. As shown in Figure 5.1, the architecture of the proposed neural network consists of four layers. Each layer contains a set of feature maps which are the results of convolutions. Applying and combining these automatically learnt operations insure the extraction of robust features, leading to the automatic detection of border positions between consecutive characters in a text image.

The first layer is the input layer E; it consists of $N_E = 3$ input maps, each of them corresponding to one color channel of the image, depending on the color space chosen (RGB, YUV, CMY, etc.). Their pixel values are normalized to the range $[-1, 1]$.

The second layer is a convolution layer C_1 which is composed of NC_1 maps. Each unit in each map is connected to a $1 \times M_1$ neighborhood in each

map of the previous layer. Furthermore, the trainable weights forming the receptive field, which play the role of convolutional mask, are forced to be equal for the entire map. This idea of weight sharing insures the extraction of the same feature over the entire map and reduces the computational cost. A trainable bias is added to the results of each convolutional mask. Each map can be considered as a feature map that has a learnt fixed feature extractor that corresponds to a pure convolution with a trainable mask, applied over the maps in the previous layer. Multiple maps lead to the extraction of multiple features.

The third layer is also a convolution layer C_2. It is composed of NC_2 maps. Each unit in each map is connected to a $1 \times M_2$ neighborhood in the maps of the previous layer. Here again, we apply the idea of weight sharing. Layer C_2 allows extracting more complex information; outputs of different feature maps of layer C_1 are fused in order to combine different features. There are different possible combination schemes that will be discussed later on.

Compared to the classical convolutional neural network architecture, the proposed architecture does not contain sub-sampling layers, which are generally used for ensuring tolerance versus pattern. Our goal being to detect the border position between consecutive characters, the exact positions of the extracted features are important, that's why there is no sub-sampling.

The last layer, as the previous two layers, is a convolution layer C_3. However, there is only one map in this layer and the convolutional mask is not a vector, but rather a matrix. Indeed, each unit in a given map of layer C_3 is connected to a $M_3 \times M_3$ neighborhood in the maps of the previous layer. Moreover, M_1, M_2 and M_3 are chosen so that the height of the map of the last layer C_3 is equal to 1. Thus, this map is a vector, where the units with high outputs correspond to a border position between two consecutive characters in the input image.

This architecture is inspired from the classical vertical projection profile analysis technique. This technique belongs to the first employed techniques for text segmentation and is applied to binary images. The vertical projection analysis consists of two steps: the first one is a simple running count of the black pixels which correspond to the black pixels vertical histogram, and the second step is the detection of the white space between successive characters. Furthermore, in our case, we are processing directly non binary images. The first step is equivalent to a feature extraction module. In our architecture, it corresponds to the hidden layers C_1 and C_2 where instead of a simple sum, we have a trainable weighted sum. Moreover, we tried an architecture without layer C_2 but the results were poor. In fact, the combination of feature maps from layer C_1 is very important to insure robustness to distortions. For the

second step, when the image is noiseless and has a high resolution, detecting the white space in the vertical histogram is sufficient to segment characters. However, if the image is noisy and distorted, more sophisticated function are required such as the second derivative ratio, the peak-to-valley function, or the column ANDing procedure [CL96]. In our architecture, this step is equivalent to the last layer C_3, which performs trainable convolution with the different maps of the layer C_2 to detect the frontier position between consecutive characters.

5.2.2 A specific implementation

The architecture of the system consists of three hidden layers where the two first ones are devoted to the extraction of relevant features for segmentation, while the last layer is concerned with the detection of frontier position between characters. A specific implementation of our character segmentation system is detailed in this section.

Layer E which contains 3 maps $(e^{(ch)})_{ch\in\{1..3\}}$ corresponding to the normalized color channels of the input image.

Layer C_1: This is the first hidden layer, it consists of $NC_1 = 6$ maps fully connected to the previous layer. Each map m have three vectors of 13 weights $(w^{(1)}_{ch,m}(u,v))_{ch\in\{1..3\},u=0,v\in\{0..12\}}$, constituting three $1 \times M_1 = 1 \times 13$ trainable kernels and corresponding to the convolutional masks operating on the three maps in the previous layer.
Thus, the values $c^{(1,m)}(x,y)$ of a given feature map $c^{(1,m)}$ are obtained by convolving the input maps $e^{(ch)}_{ch\in\{1..3\}}$ with their respective kernels $(w^{(1)}_{ch,m})_{ch\in\{1..3\}}$, adding a bias $b^{(1,m)}$ and applying an activation function Φ as shown in the following formula:

$$c^{(1,m)}(x,y) = \Phi\left(\sum_{ch=1}^{3}\sum_{v=0}^{v=12}\left(e^{(ch)}(x,y+v) \times w^{(1)}_{ch,m}(0,v)\right) + b^{(1,m)}\right) \quad (5.1)$$

The activation function here is the sigmoid function presented in chapter 3.

Layer C_2: This is the second hidden layer, it consists of $NC_2 = 27$ maps connected to the previous layer according to the scheme 'A' (cf. Table 5.1). Each map m in this layer have $NC_1 = 6$ vectors of 7 weights $(w^{(2)}_{m',m})(u,v)_{m'\in\{0..5\},u=0,v\in\{0..6\}}$, constituting the six $1 \times M_1 = 1 \times 7$ trainable kernels and corresponding to the convolutional masks operating on the six

maps in the previous layer.
The values $c^{(2,m)}(x,y)$ of the feature map $c^{(2,m)}$ are obtained by convolving the layer C_1 maps with their respective kernels $(w^{(2)}_{m',m})$, adding a bias $b^{(2,m)}$ and applying an activation function Φ as shown in the following formula:

$$c^{(2,m)}(x,y) = \Phi\left(\sum_{m'\in\xi_m}\sum_{v=0}^{v=6}\left(c^{(1,m')}(x,y+v)\times w^{(2)}_{m',m}(0,v)\right)+b^{(2,m)}\right) \quad (5.2)$$

where, ξ_m denotes the set of maps in layer C_1 to connected to the map m in layer C_2, this set depends on the combination scheme adopted; and where the activation function is the sigmoid.

Layer C_3: This is the last layer, it consists of one unique map $NC_3 = 1$ fully connected to the previous layer. This unique map has $NC_2 = 27$ matrices of 25 weights $(w^{(3)}_{m',m})(u,v)_{m'\in\{0..NC_2-1\},(u,v)\in\{-2..2\}}$, constituting a $M_3\times M_3 = 5\times 5$ trainable kernel and corresponding to the convolution mask operating on the previous layer.
The values $c^{(3,0)}(x,y)$ of the unique map $c^{(3,0)}$ are obtained by performing a weighted convolution, adding a bias $b^{3,m}$ and applying an activation function Φ. The activation function is the sigmoid function.
We have to mention here that this final map is actually one vector due to the convolution border effect. In fact, the sizes of the convolution masks are chosen so that this map has only one raw which provides us directly with the characters frontier positions. For sake of simplicity, we choose to express $c^{(3,0)}(x)$ as y is always equal to zero, as shown in the following formula:

$$\begin{aligned} c^{(3,0)}(x) = \quad \Phi \quad & \left(\sum_{m'=1}^{27}\sum_{u=-2,v=-2}^{u=2,v=2}\left(c^{(2,m')}(x+u+\frac{M_3}{2},v+\frac{M_3}{2})\right.\right. \\ & \left.\left.\times\ w^{(3)}_{m',0}(u,v)\right)+b^{(3,m)}\right) \end{aligned} \quad (5.3)$$

5.2.3 Training Phase

Data

To train such a network in a supervised mode, we need a set of color text images where the border positions between every two consecutive characters are known. To avoid the costly manual text segmentation, and because we are convinced of the good generalization ability of the proposed neural network, we choose to use synthetic images that we build automatically. Thus, a set of color text images of size $W\times H = 48\times 23$ is constructed using different

Figure 5.2: Examples of images of the training set

fonts, different text colors, different background colors, and different types of noise. Figure 5.2 shows some examples of training images.

Our training set contains 5000 RGB color text images, $N_t = 4500$ for enrolling and $N_v = 500$ for validation. We remind here the validation set is only used to evaluate the network performance after each iteration and that its images are not learnt, which means that no weights are updated as function of the error related to the validation set images. For each pattern image p of the training set, we build the desired output vector $d(x)_p$. It is a vector of width N_o elements equal to the image width W minus $(M_3 - 1)$ because of the convolution border effects. An element of this vector is set to 1 if its position correspond to a border position between two consecutive characters, and is set to -1 otherwise.

Training

The training phase was performed using the classical back-propagation algorithm with momentum modified for being used in convolutional networks as described in [LBBH98]. At each iteration, the three RGB channels of the color text image are presented to the network as inputs and the border positions between consecutive characters as the desired output vector. The weights are updated as a function of the error between the obtained output vector and the desired vector.

The objective of the network being to obtain for a given image pattern p, final values $c^{3,0}(x)_p$ equal to the desired values $d(x)_p$ where $x \in \{0..N_o - 1\}$ and $p \in \{1..N_v\}$, we classically choose to minimize the MSE (Mean Square Error) between obtained and desired values over a validation set of $N_v = 500$

images.

Similarly to the previous chapter, we define the error E_p of a training pattern p as:

$$E_p = \frac{1}{2} \sum_{x=0}^{N_o-1} (c^{(3,0)}(x)_p - d(x)_p)^2 \tag{5.4}$$

Then, the MSE over a given validation set is:

$$MSE = \frac{1}{N_v} \sum_{p=1}^{N_v} E_p \tag{5.5}$$

At each iteration, the three color channels of a given text image are presented to the network as inputs and its corresponding binary image as the desired output map. The weights update is then an iterative procedure, where their values are modified by a small variation in the opposite direction of the steepest gradient ∇E of the error E. More explicitly, the update of any matrix of weights $w^{(l)}_{m',m}$ connecting the map m' from the layer l to the map m from the layer $l-1$ is:

$$w^{(l)}_{m',m} \leftarrow w^{(l)}_{m',m} + \Delta w^{(l)}_{m',m} = w^{(l)}_{m',m} - \eta \frac{\partial E_p}{\partial w^{(l)}_{m',m}} + \alpha (\Delta w^{(l)}_{m',m})_{previous} \tag{5.6}$$

where η is the learning rate, E_p is the mean square error of a given pattern p and α is the momentum rate.

The backpropagation formulas governing the three convolution layers of our system are detailed below.

The partial derivative of a pattern error E_p according to a given weight $w^{(3)}_{m',0}(u,v)$ connecting the map m' in layer $C^{(2)}$ to the unique map 0 of layer $C^{(3)}$ at the kernel position (u,v) is:

$$\frac{\partial E_p}{\partial w^{(3)}_{m',0}(u,v)} = \sum_{x=0}^{N_o-1} [(c^{(3,0)}(x)_p - d(x)_p)(\frac{\partial c^{(3,0)}(x)_p}{\partial w^{(3)}_{m',0}(u,v)})] \tag{5.7}$$

Where,

$$\frac{\partial c^{(3,0)}(x)}{\partial w^{(3)}_{m',0}(u,v)} = c^{2,m'}(x + u + \frac{M_3}{2}, v + \frac{M_3}{2}) \times \phi'(A^{(3,0)}(x)) \tag{5.8}$$

where, ϕ' is the derivative of the activation function used in the last layer, and $A^{(3,0)}(x)$ is the activation at the unit $(x,0)$ of the map 0 of the layer $C^{(3)}$.

We define $e^{(3,0)}(x) = (c^{(3,0)}(x) - d(x))$ which is the local error, and $\delta^{(3,0)}(x) = e^{(3,0)}(x) \times \phi'(A^{(3,0)}(x))$, which corresponds to the local gradient. So,

$$\begin{aligned}\Delta w^{(3)}_{m',0}(u,v) &= -\eta \textstyle\sum_x \left(\delta^{(3,0)}(x) c^{(2,m')}(x+u+\tfrac{M_3}{2}, v+\tfrac{M_3}{2})\right) \\ &\quad +\alpha(\Delta w^{(3)}_{m',0}(u,v))_{previous} \end{aligned} \tag{5.9}$$

In a similar way, we compute the derivative of the error according to the bias. The update is then the following:

$$\Delta b^{(3,0)} = -\eta \sum_x (\delta^{(3,0)}(x)) + \alpha(\Delta b^{(3,0)})_{previous} \tag{5.10}$$

Once the weights matrices and the bias of the unique map of the last layer have been updated, the next step is to backpropagate the error to the layer $C^{(2)}$.
The local error at position (x, y) for a given map m from the layer $C^{(2)}$ is shown in the following equation :

$$e^{(2,m)}(x,y) = \sum_{(u,v)=(-2,-2)}^{(2,2)} (\delta^{(3,0)}(x-u-\frac{M_3}{2}) \times w^{(3)}_{m,0}(u,v)) \tag{5.11}$$

The local gradient is computed by multiplying the local error by the derivative of the activation function:

$$\delta^{(2,m)}(x,y) = \phi'(A^{(2,m)}(x,y)) \times \sum_{(u,v)=(-2,-2)}^{(2,2)} (\delta^{(3,0)}(x-u-\frac{M_3}{2}) \times w^{(3)}_{m,0}(u,v)) \tag{5.12}$$

Then, the weights update is:

$$\begin{aligned}\Delta w^{(2)}_{m',m}(0,v) &= -\eta \textstyle\sum_{x,y} \left(\delta^{(2,m)}(x,y) c^{(1,m')}(x,y+v)\right) \\ &\quad +\alpha \Delta w^{(2)}_{m',m}(0,v))_{previous} \end{aligned} \tag{5.13}$$

In a similar way, we compute the update of the bias:

$$\Delta b^{(2,m)} = -\eta \sum_{x,y} (\delta^{(2,m)}(x,y)) + \alpha(\Delta b^{(2,m)})_{previous} \tag{5.14}$$

Finally, we have to backpropagate the error to the first hidden layer $C^{(1)}$. The local error at position (x, y) for a given map m from the layer $C^{(1)}$ is shown in the following equation :

$$e^{(1,m)}(x,y) = \sum_{m''} \sum_{v=0}^{M_2-1} (\delta^{(2,m'')}(x,y-v) \times w^{(2)}_{m,m''}(0,v)) \tag{5.15}$$

where, the set of maps m'' represents the maps from layer C_2 connected to the map m from layer C_1, therefore, the values of m'' depend on the combination scheme adopted.
The local gradient is computed by multiplying the local error by the derivative of the activation function:

$$\delta^{(1,m)}(x,y) = \phi'(A^{(1,m)}(x,y)) \times \sum_{m''} \sum_{v=0}^{M_2-1} (\delta^{(2,m'')}(x,y-v) \times w^{(2)}_{m,m''}(0,v)) \tag{5.16}$$

Then, the weights update is:

$$\begin{aligned} \Delta w^{(1)}_{ch,m}(0,v) &= -\eta \textstyle\sum_{x,y} \left(\delta^{(1,m)}(x,y) e^{(ch)}(x,y+v)\right) \\ &\quad +\alpha \Delta w^{(1)}_{ch,m}(0,v))_{previous} \end{aligned} \tag{5.17}$$

We remind here that $e^{(ch)}_{(}ch = 1..3)$ represents the input maps and not error maps. In a similar way, we compute the update of the bias:

$$\Delta b^{(1,m)} = -\eta \sum_{x,y} (\delta^{(1,m)}(x,y)) + \alpha (\Delta b^{(1,m)})_{previous} \tag{5.18}$$

The figure 5.3 illustrates the relation of some parameters governing the backpropagation algorithm for the segmentation system.

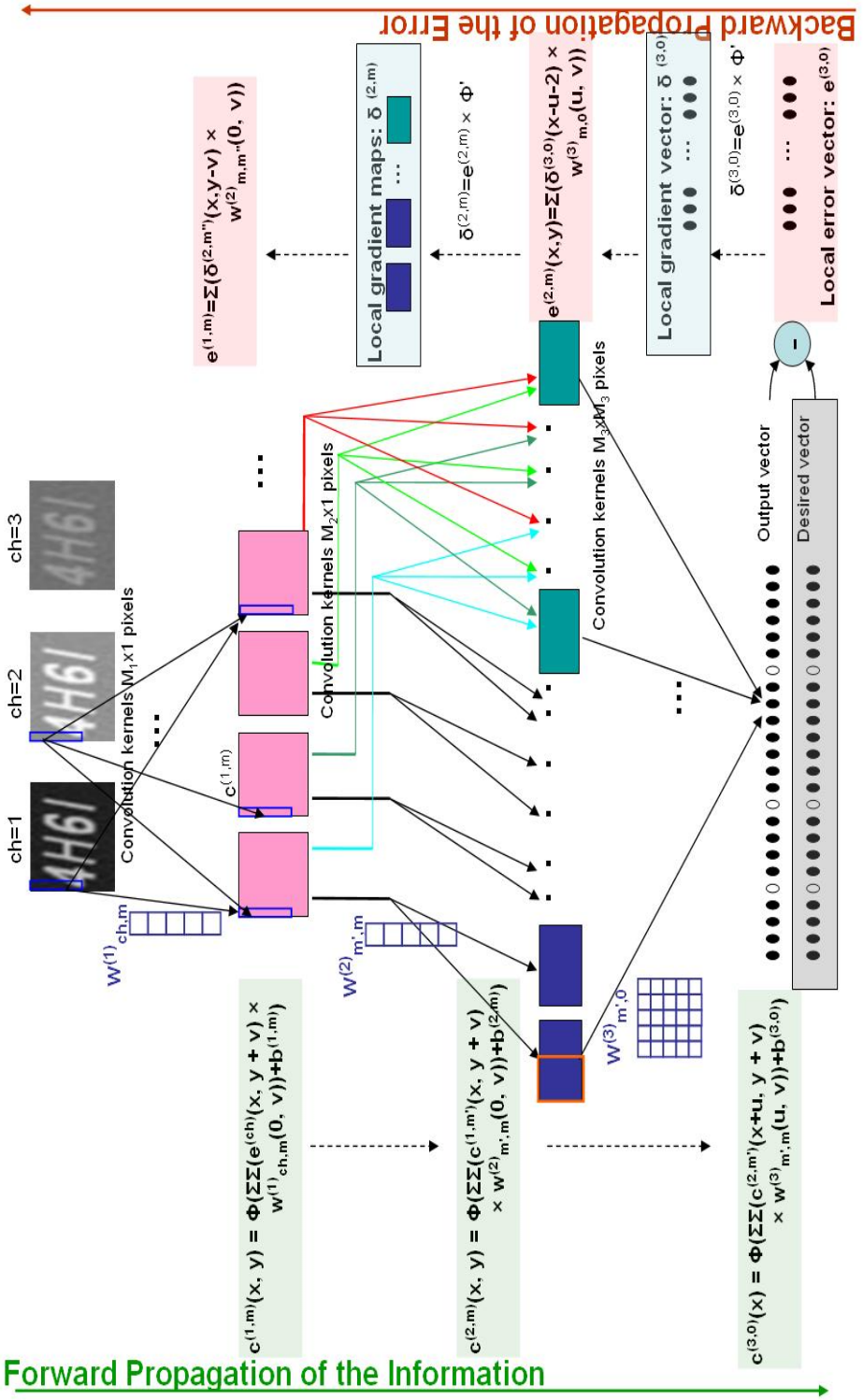


Figure 5.3: Error back-propagation in the segmentation system

5.2.4 The choice of parameters

We choose to build $NC_1 = 6$ maps in layer C_1, and $NC_2 = 27$ maps in layer C_2. Layer C_1 contains only one map. We use sigmoid activation functions for the three convolution layers C_1, C_2, and C_3. The convolution window sizes are $1 \times M_1 = 1 \times 13$, $1 \times M_2 = 1 \times 7$ and $M_3 \times M_3 = 5 \times 5$. The combination scheme between layer C_1 and layer C_2 is explained in the table 5.1.

The different parameters governing the proposed architecture, i.e., the number of maps, the combination scheme as well as the size of the receptive fields, have been experimentally chosen. We expose below some of the most relevant experiments which justify our choices. All these experiments have been running over 100 iterations and for each one, we vary one parameter and we keep the best values for the remaining parameters. We choose to evaluate the performance through the Receiver Operating Characteristic (ROC) curves. These curves measure the sensitivity versus the specificity of the system. The sensitivity or the recall rate reports the proportion of actual frontier positions which are correctly identified as such; and the specificity or the precision rate reports the proportion of non frontier positions which are identified as frontier positions. The formula of the recall rate and the precision rate are reminded here:

$$Recall = \frac{Number\ of\ correctly\ identified\ frontier\ positions}{Number\ of\ desired\ frontier\ positions} \tag{5.19}$$

$$Precision = \frac{Number\ of\ correctly\ identified\ frontier\ positions}{Number\ of\ obtained\ frontier\ positions} \tag{5.20}$$

Let us focus first on the size of the convolution windows M_1, M_2 and M_3 of layers C_1, C_2 and C_3 respectively. The first hidden layer C_1 is a feature extractor module, therefore the size of its convolution window depends on the character height. In fact, as this layer aims to extract a feature which indicates if this column belongs to the background or to the text, M_1 should approximate the character height. Provided that the characters sizes in the training set ranges approximately from 4×10 to 14×16, we choose M_1 ranging from 11 to 15. The height of the maps of the first hidden layer C_1 depends on M_1 and is equal to $H - M_1 + 1$. Given that $H = 23$ and according to the chosen range of M_1, M_2 should range from 5 to 9. Finally, the last layer C_3 is a detection layer of a height that have to be equal to one, which means that the choice of M_1 and M_2 determines M_3. Figure 5.4 shows the ROC curves of the system tested on a test set of 1000 text images for each of the size of the convolution windows experimented. This figure reports also the best couple of (recall, precision) for each category. We notice that the best result is given by the size $(M_1, M_2, M_3) = (13, 7, 5)$.

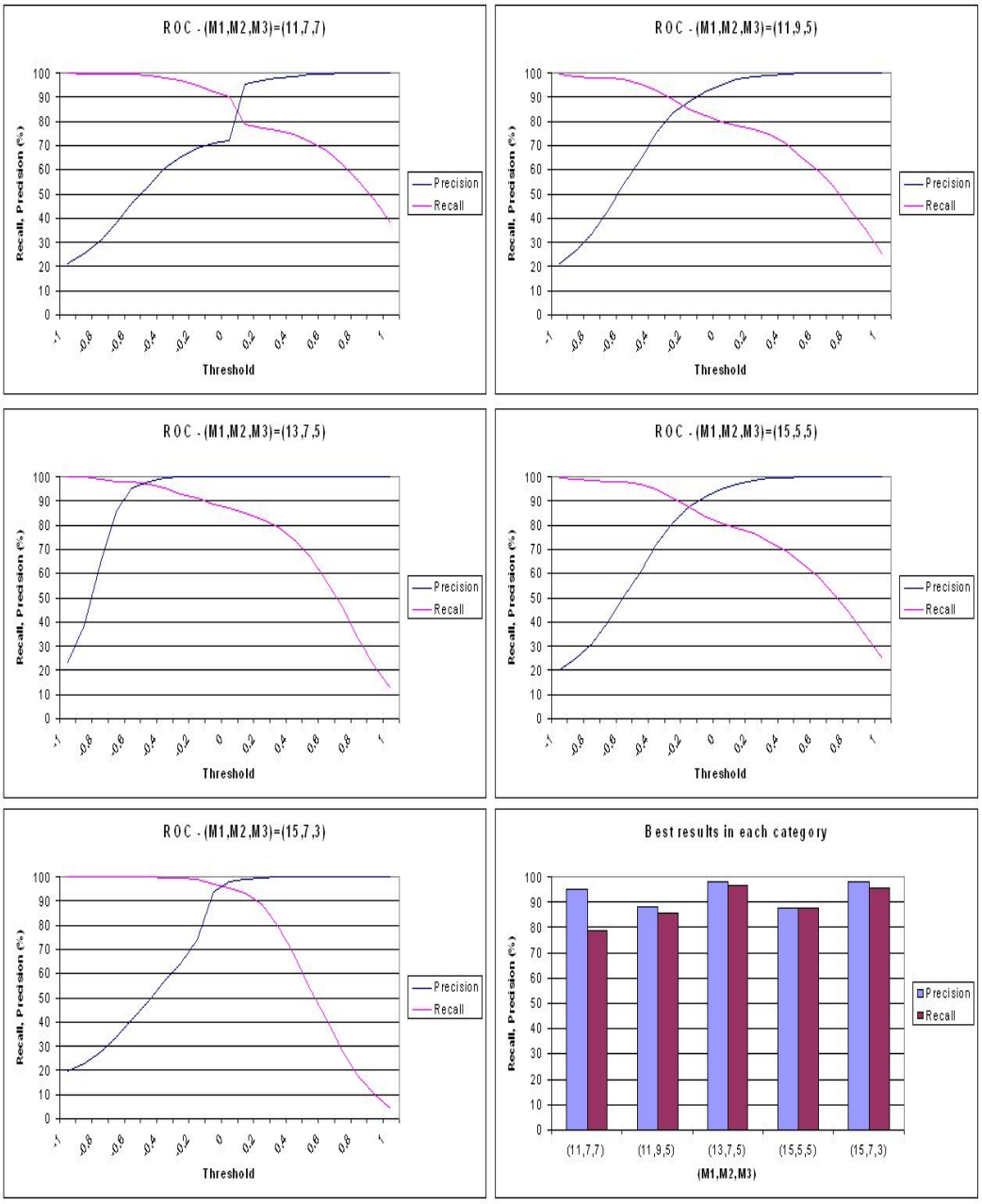


Figure 5.4: ROC curves vs. Convolution window sizes

Table 5.1: Combination Scheme A.

	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
0	X X X X X X X X X X X X X
1	X X X X X X X X X X X X
2	X X X X X X X X X X X
3	X X X X X X X X X X
4	X X X X X X X X X
5	X X X X X X X X

The second study concerns N_{C1}, the number of maps in the layer C_1. We do not study the number of maps in the layer C_2 because it is determined according to N_{C1} and the combination scheme used.
If the number of maps is small, the system does not learn well and performance is poor, whereas, if the number of maps is very big, the system may over learn and the performance is also poor. Figure 5.5 shows the ROC curves of the system tested on a test set of 1000 text images for each of the number of maps experimented. This figure reports also the best couple of (recall, precision) for each category. We notice that the best result is given by $N_{C1} = 6$.

Finally, we propose to study some combination schemes governing the connections between layer C_1 and layer C_2:

- SCHEME A: is explained in table 5.1, it shows two kinds of feature maps, the first category is generated from only one previous feature map and it contains $2 \times N_{C1}$ and the second category is generated from the combination of two previous feature maps and it contains $\frac{N_{C1} \times (N_{C1}-1)}{2}$.

- SCHEME B: is explained in table 5.2. It corresponds to the different possible combination of two maps from layer C_1. Therefore, $N_{C2} = \binom{2}{N_{C1}} = \frac{N_{C1}!}{(N_{C1}-2)! \times 2!}$

- SCHEME C: is explained in table 5.3. The first six C_2 feature maps take inputs from every contiguous subset of three feature maps in C_1. The next six take input from every contiguous subset of four feature maps in C_1. The next three take input from some discontinuous subsets of four feature maps in C_1. Finally, the last one takes input from all C_1 feature maps.

- SCHEME D:is fully connected and $N_{C2} = 27$

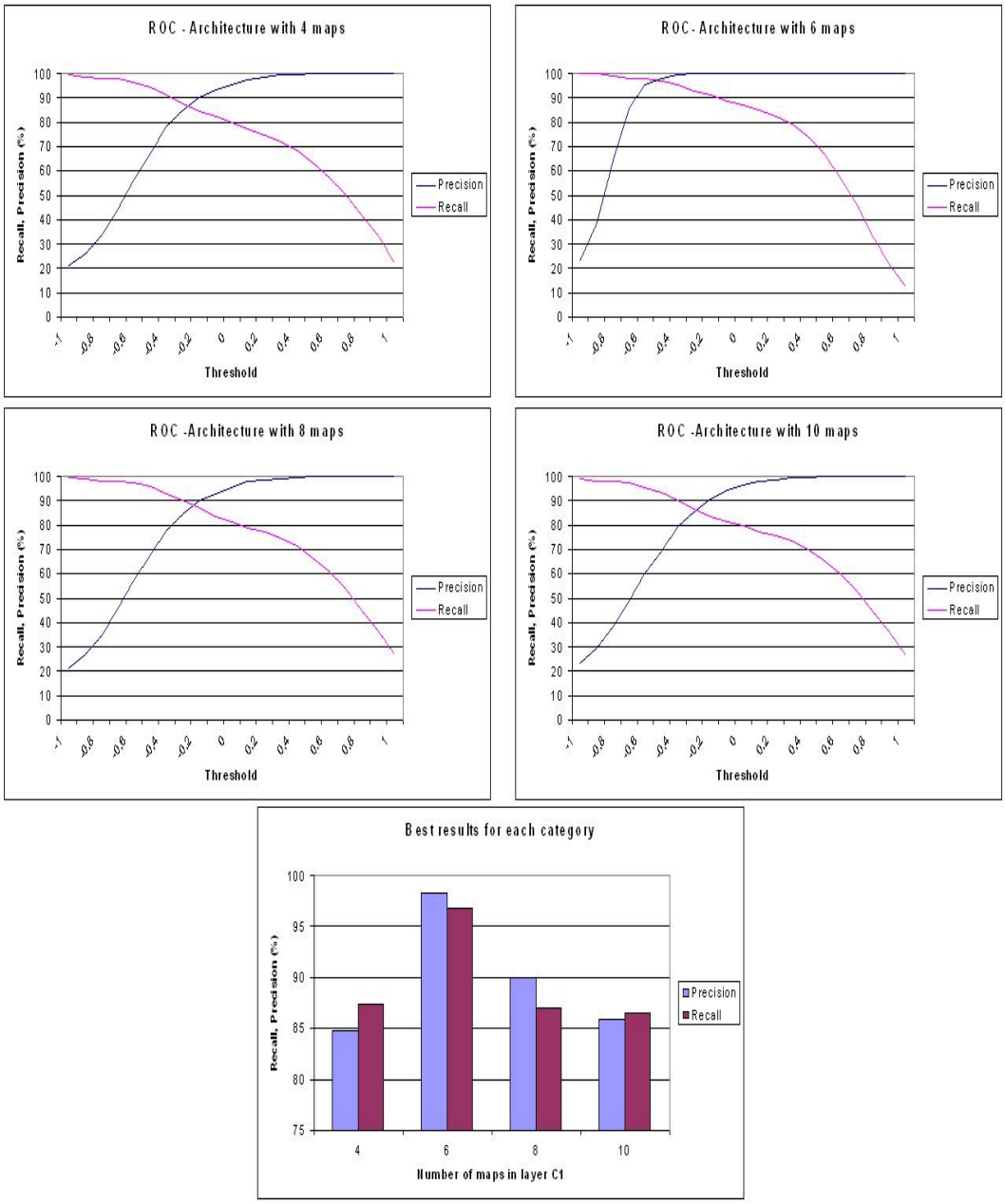


Figure 5.5: ROC curves vs. Number of maps in layer C1

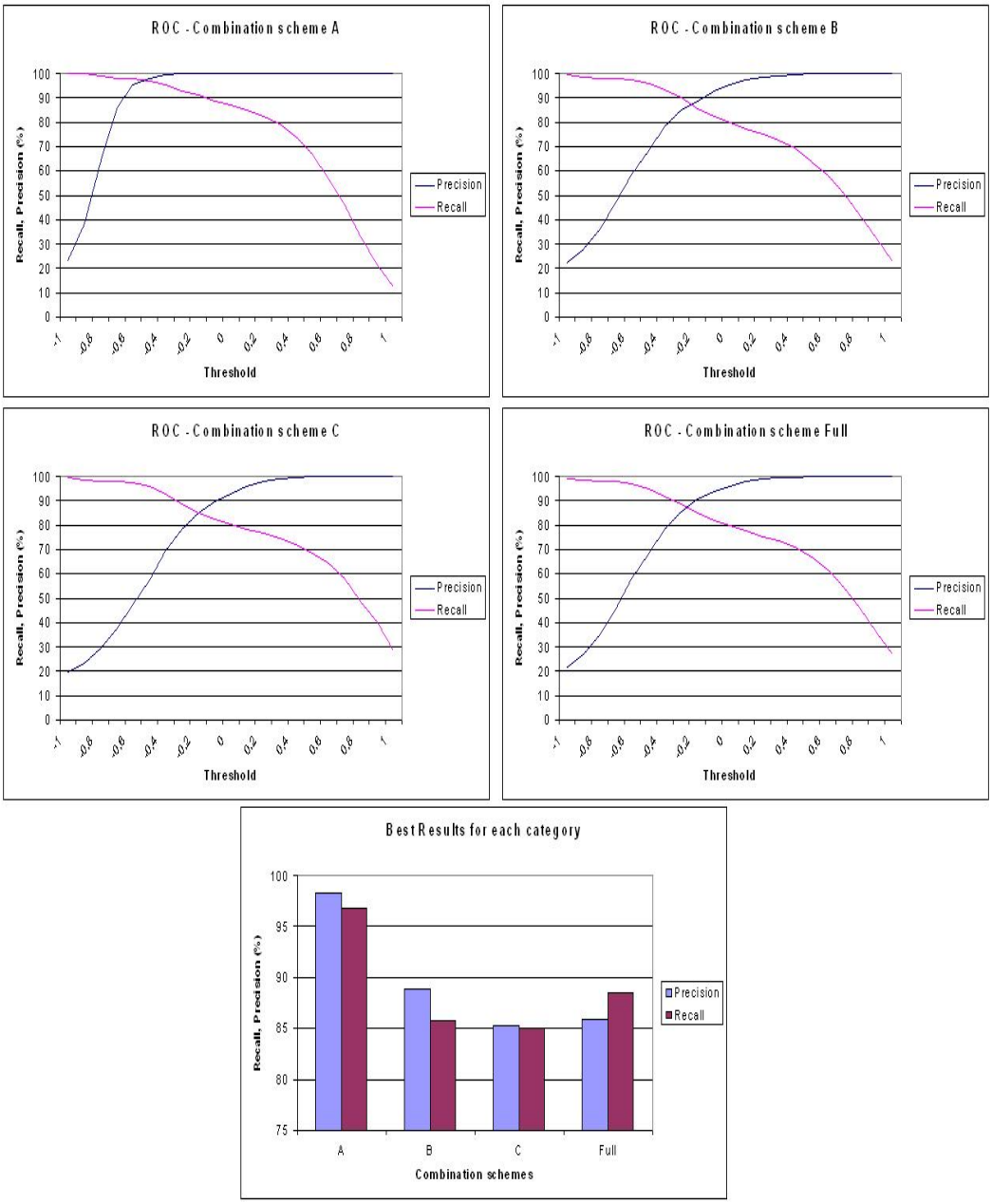


Figure 5.6: ROC curves vs. combination schemes

Table 5.2: Combination Scheme B.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	X				X				X				X		X
1	X	X				X				X				X	
2		X	X		X		X				X				
3			X	X		X		X	X						
4				X	X		X			X		X			
5					X			X			X			X	X

Table 5.3: Combination Scheme C.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

Figure 5.6 shows the ROC curves of the system tested on a test set of 1000 text images for each of the scheme experimented. This figure reports also the best couple of (recall, precision) for each category. We notice that the best result is given by scheme A. Moreover, we were expecting that scheme D will get the worst performance as it is a full connection and it does not break the symmetry in the network, but surprisingly this was not the case. This is may be due to the number of maps in layer C_2 in scheme D which is bigger than the one in schemes B and C.

5.2.5 Testing Phase

Once the training is achieved, the system is ready to segment color text images a follows:

1. Resize the text image to the retina height, while preserving the aspect ratio (the size of the training input images are called retina size).

2. Feed the network with the resized text image as input

3. Retrieve the output vector of the network. The units of this vector with high outputs correpond to the border positions detected by the system. So, save then the column indexes of these border positions detected.

4. Crop the text image according to these column indexes

5. Each cropped symbol is considered as a character image.

As in the CTB system, here again the width of the image must not be equal to the retina width, because of the sharing weights property of this network. In fact, for each of the three layers, there is only one convolution mask per map which corresponds to the matrix of weights. This convolution mask can be applied as much as necessary according to the width of the image processed.

Once character images are available, a character recognition system can be applied. We choose to use a character recognition system based also on convolutional neural network, that we proposed recently [SG07a] and which yields very good results compared to the state of the art systems.

5.3 Experimental Results

To test the performance of our method, we use two databases. The first one contains 5000 synthetic text color images, where the exact positions of the different characters are known. This database is used to evaluate the performance of our segmentation system according to noise. This database includes five categories of 1000 images with a gaussian noise variance ranging from 3 to 11 by a step of 2. For this evaluation, it is easy to compute the recall and the precision criteria since the character positions are known. Given that a border position between two consecutive characters is not unique, we allow a margin of n columns. In other words, if the desired fontier position is P and the system find a border position beetween $P - n$ and $P + n$, then this position is considered as correct. A margin of $n = 2$ seems to be acceptable.

Figure 5.6 shows the ROC curves corresponding to each noise category. This figure reports also the best couple of (recall, precision) for each category. The variation of recall and precision does not exceed 2% and remains always above the 80% which illustrates the robustness of the system against noise variation.

The second test database contains 52 real images collected from video frames and web pages. Figure 5.8 shows some examples from the test images data set. We apply the proposed segmentation system and we manually evaluate the number of corrected segments. In the test set, we have 469 characters. We have found 418 correctly segmented characters, which means that our algorithm reach a correct segmentation rate of 89.12%.

After evaluating the segmentation rate, we evaluate also on this database the recognition rates of the overall system combining the text segmentation system presented here and the character recognition system detailed in the next chapter. We compare this overall system with some text recognition

schemes based on the combination of a binarization technique and a classical OCR. We use the following four state of the art binarization techniques that we combine with two OCRs (Tesseract which is a public software and Abby FineReader 8.0 which is a commercial software):

1. the Niblack method [Nib86] computes a threshold based on the mean and the standard deviation of each block in the image, and then binarize the image.

2. the Sauvola method [SSHP97] is also based on the mean and the standard deviation to binarize the text image.

3. the Lienhart method [LW02] consists in choosing the intensity separating value halfway between the intensity of the text color and the background color as a binarization threshold.

4. the CTB method [SG07b] is a our binarization method based on a specific convolutional neural network presented in the previous chapter.

Figure 5.9 shows the corresponding recognition rates.

While our method outperforms the classical techniques [Nib86], [SSHP97], [LW02], it does not outperform the CTB combined with the commercial OCR, FineReader. Nevertheless, we think that the results are encouraging and improvement could be achieved on the overall scheme by considering some linguistic analysis as most of the classical OCR systems. This will be considered in a future work.

5.4 Conclusion

In this chapter, we have proposed a novel approach based on specific convolution neural network architecture designed for complex color text image segmentation. This approach is a one step approach which is robust to noise, low resolution and complex background. Evaluated on a set of text images collected from video frames and web pages, our method gives encouraging results. Now, that we have a system able to produce character images from a text image, the next step will focus on character recognition system.

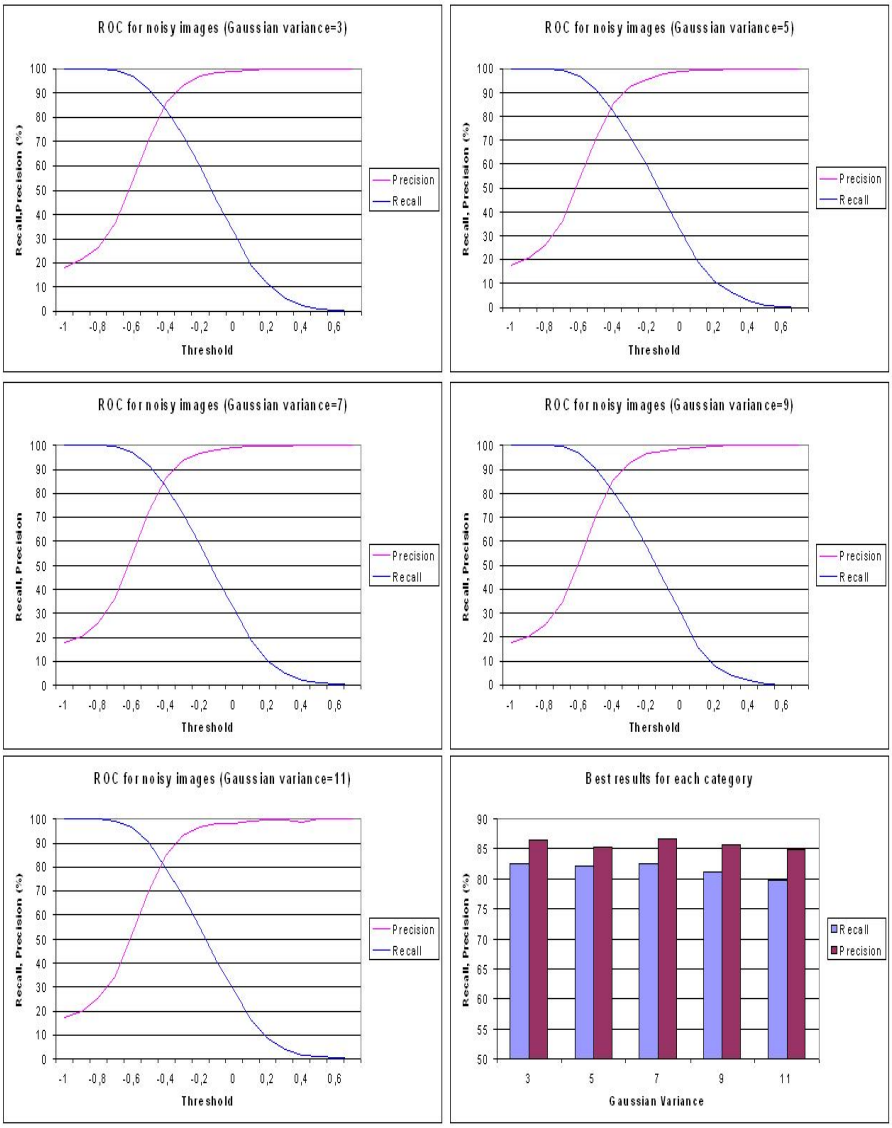


Figure 5.7: Performance of the segmentation system as function of noise

Figure 5.8: Some segmentation test examples

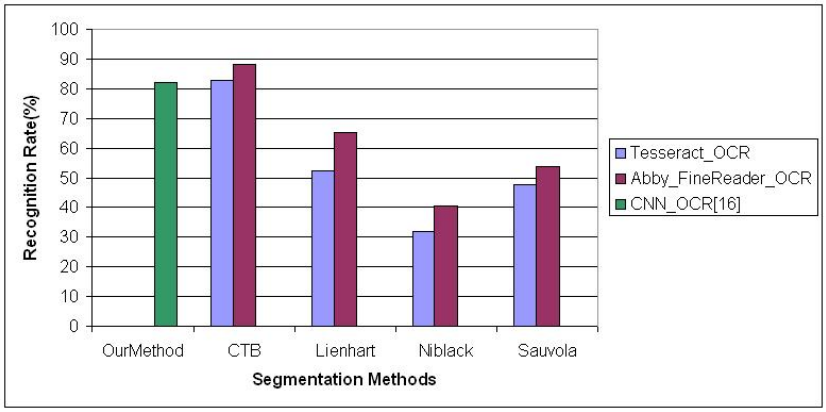


Figure 5.9: Comparison of our method with other state of the art techniques

Chapter 6

Character Recognition in Text Images

Ce qui est affirmé sans preuve
peut être nié sans preuve.

What is stated without proof
can be denied without proof.
[Euclide](-325 , -265)

6.1 Introduction

In the previous chapter, we presented a segmentation system, which for a given text image outputs the frontier positions between consecutive characters and so we can extract character images ready to be recognized.

In this chapter, we propose an automatic recognition scheme for natural color character images, based on supervised learning, without applying any pre-processing like binarization, without making any assumptions and without using tunable parameters. Moreover, our system makes a direct use of color information and insures robustness to noise, to complex backgrounds and to luminance variations.

The remainder of this chapter is organized as follows. Section 6.2 describes in detail the architecture of the proposed neural network. It explains also the training process. Experimental results are reported in Section 6.3. Conclusions are drawn in Section 6.4.

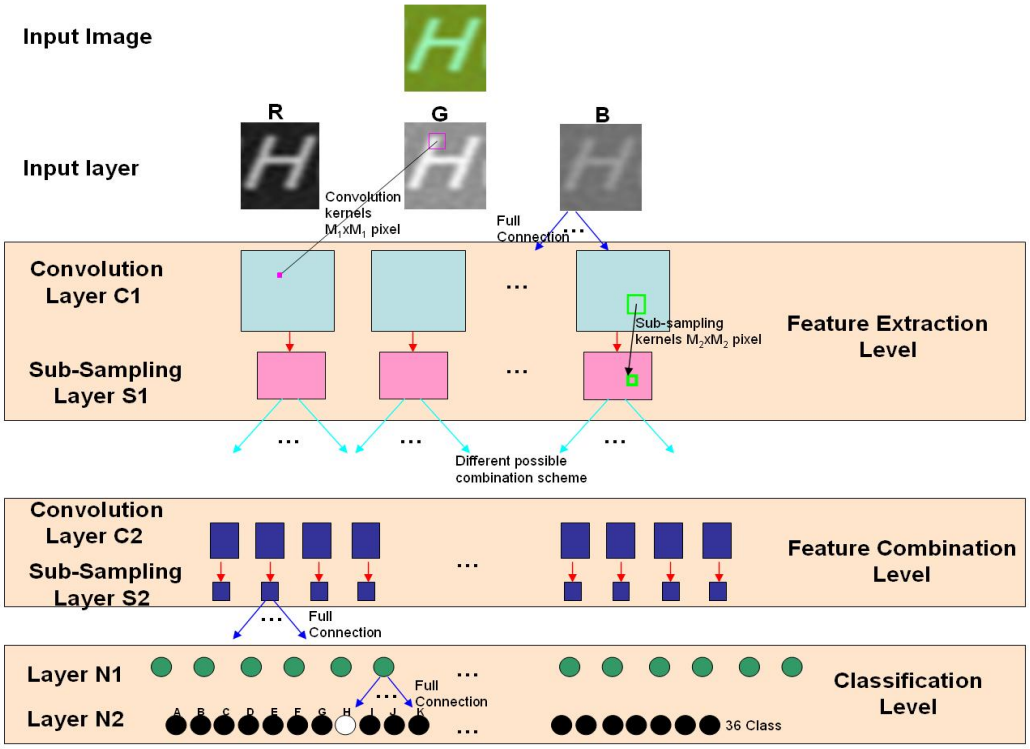


Figure 6.1: Architecture of the Network

6.2 The character recognition system

6.2.1 System Architecture

The proposed neural architecture is based on convolutional neural networks.

As shown in Figure 6.1, the proposed network consists of seven different heterogeneous layers. Each layer contains feature maps which are the results of convolution, sub-sampling, or neuron unit operations. Applying and combining these automatically learnt operations ensure the extraction of robust features, leading to the automatic recognition of characters in natural images.

The first layer is the input layer E; it consists of $N_E = 3$ input maps, each of them corresponding to one color channel of the image, depending on the color space (RGB, YUV, CMY, etc.). Their pixel values are normalized to the range $[-1, 1]$. We can distinguish then three main levels:

LEVEL 1: Feature extraction level, relying on the C_1 and S_2 layers.

The layer C_1 extracts some specific features (oriented edges, corners, end points...) directly from the three color channels. In fact, this layer is composed of NC_1 maps. This is a convolution layer where each unit in each map is connected to a $M_1 \times M_1$ neighborhood (biological local receptive field) in each of the color channels of the input layer. Furthermore, the trainable weights (convolutional mask) forming the receptive field, are forced to be equal for the entire map (weight sharing). A trainable bias is added to the results of each convolutional mask. Thus, each map can be considered as a feature map that has a learnt fixed feature extractor that corresponds to a pure convolution with a trainable mask, applied over the channels in the input layer. Multiple maps lead to the extraction of multiple features.

Once a feature is extracted its exact position is no longer important; only its relative position to other feature is relevant. Therefore, each map of the third layer S_2 results from local averaging and sub-sampling operations on a corresponding map in the previous layer C_1. So, the layer S_2 is composed of $NS_2 = NC_1$ feature maps. We use this sub-sampling layer to reduce by two the spatial resolution which reduces the sensitivity to shifts, distortions and variations in scale and rotation.

Level 2: Feature combination level, relying on the C_3 and S_4 layers.

Layer C_3 enables to extract more complex information; outputs of different feature maps of layer S_2 are fused in order to combine different features. There are many possible combination schemes that will be discussed later on. This is the second convolution layer, each unit in each map is connected to a $M_3 \times M_3$ neighborhood in some of the maps of the layer S_2 according to the combination scheme adopted. As in layer C_1, the trainable weights forming the receptive field are forced to be equal for the entire map. A trainable bias is added to the results of each convolutional mask.

As in the first level, in this second level also, we consider a sub-sampling layer S_4, where each map results from local averaging and sub-sampling operations applied on a corresponding map in the previous layer C_3. Indeed, this progressive reduction of spatial resolution compensated by a progressive increase of the richness of the representation, which corresponds to the number of feature maps, enables a large degree of invariance to geometric transformations of the input.

Level 3: Classification level, relying on the N_5 and N_6 layers.

Each of them is a fully connected layer and contains classical neural units. The choice of the number of units in N_5 is empirical; whereas the number of units in N_6 depends on the number of classes to be recognized. If we consider the alphanumerical patterns, we will get 62 classes (26 lower case

characters, 26 upper case characters and 10 digits). In order to reduce the number of output classes and consequently the number of neural weights to be learnt in between layers N_5 and N_6, we propose to use only 36 classes, where corresponding upper case and lower case characters are fused. This is made possible thanks to the strong generalization abilities of this network.

6.2.2 A specific Implementation

The architecture of the system consists of six hidden layers belonging to three different levels: feature extraction, feature combination and classification level. A specific implementation of the character recognition system is detailed in this section.

Layer E which contains 3 maps $(e^{(ch)})_{ch\in\{1..3\}}$ corresponding to the normalized color channels of the input image of size 48×48.

Layer C_1: This is the first hidden layer, it consists of $NC_1 = 6$ maps fully connected to the previous layer. Each map m have three matrices of 25 weights $(w^{(1)}_{ch,m}(u,v))_{ch\in\{1..3\},(u,v)\in\{-2..2\}}$, constituting three $M_1 \times M_1 = 5 \times 5$ trainable kernels and corresponding to the convolutional masks operating on the three maps in the previous layer. Therefore, the size of a given map in this layer is 44×44 because of the convolution border effect.
The values $c^{(1,m)}(x,y)$ of a given feature map $c^{(1,m)}$ are obtained by convolving the input maps $e^{(ch)}_{ch\in\{1..3\}}$ with their respective kernels $(w^{(1)}_{ch,m})_{ch\in\{1..3\}}$, adding a bias $b^{(1,m)}$ and applying an activation function Φ as shown in the following formula:

$$c^{(1,m)}(x,y) = \Phi\left(\sum_{ch=1}^{3}\sum_{u=-2,v=-2}^{u=2,v=2}\left(e^{(ch)}(x+u+\frac{M_1}{2}, y+v+\frac{M_1}{2}) \times w^{(1)}_{ch,m}(u,v)\right) + b^{(1,m)}\right) \tag{6.1}$$

The activation function here is the identity function.

Layer S_2: This is the second hidden layer, it consists of $NS_2 = 6$ maps. Each one is connected to one corresponding map from the previous layer. So, each map m in this layer have only one trainable weight $(w^{(2)}_{m,m})(0)$, which is a coefficient applied to the sub-sampling mask operating on the map m of the previous layer C_1. Therefore, the size of a given map in this layer is

22×22.
The values $s^{(2,m)}(x,y)$ of the feature map $s^{(2,m)}$ are obtained by performing a weighted sub-sampling of factor two, adding a bias $b^{(2,m)}$ and applying an activation function Φ as shown in the following formula:

$$s^{(2,m)}(x,y) = \Phi \left(\left(w^{(2)}_{m,m}(0) \sum_{u=0,v=0}^{u=1,v=1} c^{(1,m)}(2x+u, 2y+v) \right) + b^{(2,m)} \right) \tag{6.2}$$

The activation function here is the sigmoid function.

Layer C_3: This is the third hidden layer, it consists of $NC_3 = 16$ maps connected to the previous layer according to the scheme 'C' (cf. Figure 6.3). Each map m in this layer have $card(\xi_{3,m})$ matrices of 25 trainable weights $(w^{(3)}_{m',m})(u,v)_{m' \in \{0..5\},(u,v) \in \{-2..2\}}$, where $\xi_{3,m}$ is the set of maps in layer S_2 connected to map m in layer C_3 and $card(\xi_{3,m})$ is the cardinal of this set, this set depends on the combination scheme chosen. Each matrix constitutes an $M_1 \times M_1 = 5 \times 5$ trainable kernel and corresponds to the convolutional mask operating on one map from the $\xi_{3,m}$ set of maps belonging to the previous layer. Therefore, the size of a given map in this layer is 18×18 because of the convolution border effect.
The values $c^{(3,m)}(x,y)$ of the feature map $c^{(3,m)}$ are obtained by convolving the set of maps $\xi_{3,m}$ from layer S_2 with their respective kernels $(w^{(3)}_{m',m})$, adding a bias $b^{(2,m)}$ and applying an activation function Φ which is the identity function, as shown in the following formula:

$$c^{(3,m)}(x,y) = \Phi \left(\sum_{m' \in \xi_{3,m}} \sum_{u=-2,v=-2}^{u=2,v=2} \Big(s^{(2,m')}(x+u+2, y+v+2) \times w^{(3)}_{m,m'}(u,v) \Big) + b^{(3,m)} \right) \tag{6.3}$$

Layer S_4: This is the fourth hidden layer, it consists of $NS_4 = NC_3 = 16$ maps. Each one is connected to one corresponding map from the previous layer. So, each map m in this layer have only one trainable weight $(w^{(4)}_{m,m})(0)$, which is a coefficient applied to the sub-sampling mask operating on the map m of the previous layer C_3. Therefore, the size of a given map in this layer is 9×9.
The values $s^{(4,m)}(x,y)$ of the feature map $s^{(4,m)}$ are obtained by performing

a weighted sub-sampling of factor two, adding a bias $b^{(4,m)}$ and applying an activation function Φ which is a sigmoid here, as shown in the following formula:

$$s^{(4,m)}(x,y) = \Phi \left(\left(w^{(4)}_{m,m}(0) \sum_{u=0,v=0}^{u=1,v=1} c^{(3,m)}(2x+u, 2y+v) \right) + b^{(4,m)} \right) \quad (6.4)$$

Layer N_5: This is the sixth hidden layer, it consists of 120 neurons. Each neuron is fully connected to the maps of the previous layer. So, each unit u has a vector of $16 \times 9 \times 9 = 1296$ weights $w^{(5)}_{m',u}$. The output of each unit is computed classically as follow:

$$n^{(5,u)} = \Phi \left(\sum_{m'=1}^{16} \sum_{(x,y)=(0,0)}^{(8,8)} \left(w^{(5)}_{m',u}(x+9y) \times s^{(4,m')}(x,y) + b^{(5,u)} \right) \right) \quad (6.5)$$

The activation function here is the sigmoid function.

Layer N_6: This is the last layer, it consists of 36 neurons corresponding to the 36 character classes. Each neuron is fully connected to all the neurons of the previous layer. So, each unit u has a vector of 120 weights $w^{(6)}_{u',u}$. The output $n^{(6,u)}$ of a given unit u is computed classically as follow:

$$n^{(6,u)} = \Phi \left(\sum_{u'=1}^{120} \left(w^{(6)}_{u',u} \times n^{(5,u')} + b^{(6,u)} \right) \right) \quad (6.6)$$

The activation function here is the sigmoid function. For seek of simplicity, we denote $\{O_h\}_{h=1..N_{Class}} = \{n^{(6,u)}\}_{u=1..N_{Class}}$ the output vector.

6.2.3 Training Phase

Database

We believe that using a public database is important to contribute to the clear understanding of the current state of the art. Therefore, we choose to use the ICDAR 2003 database, which can be downloaded from:

"http://algoval.essex.ac.uk/icdar/Datasets.html".

ICDAR 2003 proposed a competition named robust reading [LPS+05] to refer to text images that are beyond the capabilities of current commercial

OCR packages. They chose to break down the robust reading problem into three sub-problems, and run competitions for each of them, and also a competition for the best overall system. The sub-problems are text locating, character recognition and word recognition. Due to the complexity of the database, contestants participated only to the text locating contest.

In this chapter, we propose to perform character recognition on the ICDAR 2003 single character database. This database is divided into three subsets: a train subset (containing 6185 images), a test subset (containing 5430 images), and a sample subset (containing 854 images). These character images are of different size (5x12, 36x47, 206x223...), different fonts, different colors, and present different kinds of distortion.

We used the train and the test subsets (a total of 11615 images) for training our network. Moreover, to enhance the robustness of the system, we increased the number of images in the training set by adding the corresponding negative images, and corresponding noisy images (we add Gaussian noise) of the original dataset to the final training set. Therefore, the final training set contains 34845 images.

We tested the performance of our system on the sample subset of 698 images. Actually, the sample subset contains 854 character images, but some of them are not recognizable even by humans, therefore we eliminated these cases.

Figure 6.2: Examples of images of the training set

Training

As mentioned before, we use 34845 color character scene images, $N_t = 30000$ in the training set and $N_v = 4845$ in the validation set. In fact, to avoid overfitting the training data and to increase the generalization ability of the system, a part of the whole training set is kept as validation set. This validation set is used to evaluate the network performance through iterations, as explained later on.

The training phase was performed using the classical back-propagation algorithm with momentum modified for being used in convolutional networks as described in [LBBH98] and it consists of the following steps:

1. Construct the desired output vector $\{D_h\}_{h=1..N_{Class}}$: for a given character image, belonging to class h, this desired output vector is constructed by setting its h^{th} element to 1, and setting the remaining elements to -1.

2. Compute the actual output vector $\{O_h\}_{h=1..N_{Class}}$: the character image is presented to the network as input, the last layer output vector represents the actual output vector.

3. Update weights: the weights are updated by backpropagation of the error between the actual output vector and the desired output vector.

4. Repeat step 1 until 3 for the N_t character images of the training set.

5. Compute the Mean Square Error (MSE) over the validation set: for every character images of the validation set, repeat step 1 and 2, and then compute the MSE between the actual output vectors $\{O_{h,k}\}_{h=1..N_{Class},k=1..N_v}$ and the desired output vectors $\{D_{h,k}\}_{h=1..N_{Class},k=1..N_v}$ as follow:

$$MSE = \frac{1}{N_v \times N_{Class}} \sum_{k=1}^{N_v} \sum_{h=1}^{N_{Class}} (O_{h,k} - D_{h,k})^2 \tag{6.7}$$

6. Save the weights values if MSE is decreasing.

7. Repeat steps 1 until 6, until the desired number of iterations is reached.

At each iteration, the three color channels of a given character image are presented to the network as inputs, the desired output vector is constructed according to the character class. The weights update is then an iterative procedure, where their values are modified by a small variation in the opposite direction of the steepest gradient ∇E of the error E between the desired vector and the actual vector. More explicitly, the update of any matrix of weights $w^{(l)}_{m',m}$ connecting the map m' from the layer l to the map m from the layer $l-1$ is:

$$w^{(l)}_{m',m} \leftarrow w^{(l)}_{m',m} + \Delta w^{(l)}_{m',m} = w^{(l)}_{m',m} - \eta \frac{\partial E_p}{\partial w^{(l)}_{m',m}} + \alpha(\Delta w^{(l)}_{m',m})_{previous} \tag{6.8}$$

where η is the learning rate, E_p is the mean square error of a given pattern p and α is the momentum rate.

The backpropagation formulas governing the different hidden layers of our system are detailed below.

Backpropagation at the last Level:
The error E_p of a training image p is:

$$E_p = \frac{1}{2} \sum_{h=1}^{N_{Class}} ((O_h)_p - (D_h)_p)^2 \tag{6.9}$$

Therefore, the partial derivative of a pattern error E_p according to a given weight $w^{(6)}_{u',u}$ connecting the unit u' in layer $N^{(5)}$ to the unit u in layer $N^{(6)}$ is:

$$\frac{\partial E_p}{\partial w^{(6)}_{u',u}} = ((O_u)_p - (D_u)_p)(\frac{\partial (O_u)_p}{\partial w^{(6)}_{u',u}}) \tag{6.10}$$

Where,

$$\frac{\partial (O_u)_p}{\partial w^{(6)}_{u',u}} = n^{(5,u')} \times \phi'(A^{(6,u)}) \tag{6.11}$$

where, ϕ' is the derivative of the activation function used in the last layer, and $A^{(6,u)}$ is the activation at the unit u of the layer $N^{(6)}$.

We define $e^{(6,u)} = (O_u - D_u))$ which is the local error, and $\delta^{(6,u)} = e^{(6,u)} \times \phi'(A^{(6,u)})$, which corresponds to the local gradient. We omit p for seek of simplicity.
So,

$$\Delta w^{(6)}_{u',u} = -\eta \left(\delta^{(6,u)} n^{(5,u')} \right) + \alpha (\Delta w^{(6)}_{u',u})_{previous} \tag{6.12}$$

In a similar way, we compute the derivative of the error according to the bias. The update is then the following:

$$\Delta b^{(6,u)} = -\eta \delta^{(6,u)} + \alpha (\Delta b^{(6,u)})_{previous} \tag{6.13}$$

Once the weights and the bias of the last layer have been updated, the next step consists on backpropagating the error to the layer $N^{(5)}$.
The local error for a unit u from the layer $N^{(5)}$ is shown in the following equation :

$$e^{(5,u)} = \sum_{u''=0}^{N_{Class}} (\delta^{(6,u'')} \times w^{(6)}_{u,u''}) \tag{6.14}$$

The local gradient is computed by multiplying the local error by the derivative of the activation function:

$$\delta^{(5,u)} = \phi'(A^{(5,u)}) \sum_{u''=0}^{N_{Class}} (\delta^{(6,u'')} \times w^{(6)}_{u,u''}) \tag{6.15}$$

Then, the update of a weight at layer N_5 connecting the unit r at the position (x, y) from the map m' from layer S_4 to a unit u from layer N_5 is:

$$\Delta w^{(5)}_{m',u}(r) = -\eta\delta^{(5,u)}s^{(4,m')}(x,y) + \alpha\Delta w^{(5)}_{m',u}(r))_{previous} \tag{6.16}$$

In a similar way, we compute the update of the bias:

$$\Delta b^{(5,u)} = -\eta\delta^{(5,u)} + \alpha(\Delta b^{(5,u)})_{previous} \tag{6.17}$$

Backpropagation at the second Level:
The local error at position (x, y) for a given map m from the layer $S^{(4)}$ is shown in the following equation :

$$e^{(4,m)}(x,y) = \sum_{u''=0}^{N_{N5}-1}(\delta^{(5,u'')} \times w^{(5)}_{m,u''}(x+9y)) \tag{6.18}$$

The local gradient is computed by multiplying the local error by the derivative of the activation function:

$$\delta^{(4,m)}(x,y) = \phi'(A^{(4,m)}(x,y)) \times \sum_{u''=0}^{N_{N5}-1}(\delta^{(5,u'')} \times w^{(5)}_{m,u''}(x+9y)) \tag{6.19}$$

Then, the update of the weight $w^{(4)}_{m',m}$ connecting each four element from the map m' from layer C_3 to one element from the map m from layer $S_4$is:

$$\begin{aligned}\Delta w^{(4)}_{m',m} &= -\eta \textstyle\sum_{x,y}\left(\delta^{(4,m)}(x,y)\sum_{(p,q)\in\{0,1\}} c^{(3,m')}(2x+p,2y+q)\right) \\ &\quad +\alpha\Delta w^{(4)}_{m',m})_{previous}\end{aligned} \tag{6.20}$$

In a similar way, we compute the update of the bias:

$$\Delta b^{(4,m)} = -\eta\sum_{x,y}(\delta^{(4,m)}(x,y)) + \alpha(\Delta b^{(4,m)})_{previous} \tag{6.21}$$

The local error at position (x, y) for a given map m from the layer $C^{(3)}$ is shown in the following equation :

$$e^{(3,m)}(x,y) = \delta^{(4,m)}(x/2, y/2) \times w^{(4)}_{m,m} \tag{6.22}$$

The local gradient is computed by multiplying the local error by the derivative of the activation function:

$$\delta^{(3,m)}(x,y) = \phi'(A^{(3,m)}(x,y)) \times \delta^{(4,m)}(x/2, y/2) \times w^{(4)}_{m,m} \tag{6.23}$$

Then, the update of the weight matrices $w^{(3)}_{m',m}(u,v)$ where $(u,v) \in \{-2..2\}$ connecting the map m' from layer S_2 to the map m from layer $C_3$is:

$$\Delta w^{(3)}_{m',m}(u,v) = -\eta \sum_{(x,y)=(0,0)}^{(18,18)} \left(\delta^{(3,m)}(x,y) s^{(2,m')}(x+u+\tfrac{M_3}{2}, y+v+\tfrac{M_3}{2})\right) + \alpha \Delta w^{(4)}_{m',m})_{previous} \quad (6.24)$$

In a similar way, we compute the update of the bias:

$$\Delta b^{(3,m)} = -\eta \sum_{x,y} (\delta^{(3,m)}(x,y)) + \alpha (\Delta b^{(3,m)})_{previous} \quad (6.25)$$

Backpropagation at the first Level:
The local error at position (x,y) for a given map m from the layer $S^{(2)}$ is shown in the following equation :

$$e^{(2,m)}(x,y) = \sum_{m'' \in \xi_m^{(inv)}} \sum_{(u,v)=(-2,-2)}^{(2,2)} (\delta^{(3,m'')}(x-u-\frac{M_1}{2}, y-v-\frac{M_1}{2}) \times w^{(3)}_{m,m''}(u,v)) \quad (6.26)$$

where, $\xi_m^{(inv)}$ is the set of maps m'' from layer C_3 which have connection with the map m from layer S_2. The local gradient is computed by multiplying the local error by the derivative of the activation function:

$$\delta^{(2,m)}(x,y) = \phi'(A^{(2,m)}(x,y)) \sum_{m'' \in \xi_m^{(inv)}} \sum_{(u,v)} \Big(\delta^{(3,m'')}(x-u-\tfrac{M_1}{2}, y-v-\tfrac{M_1}{2}) \times w^{(3)}_{m,m''}(u,v) \Big) \quad (6.27)$$

Then, the update of the weight $w^{(2)}_{m,m}$ connecting each four element from the map m from layer C_1 to one element from the map m from layer S_2 is:

$$\Delta w^{(2)}_{m,m} = -\eta \sum_{x,y} \left(\delta^{(2,m)}(x,y) \sum_{(p,q)\in\{0,1\}} c^{(1,m)}(2x+p, 2y+q)\right) + \alpha \Delta w^{(2)}_{m,m})_{previous} \quad (6.28)$$

In a similar way, we compute the update of the bias:

$$\Delta b^{(2,m)} = -\eta \sum_{x,y} (\delta^{(2,m)}(x,y)) + \alpha (\Delta b^{(2,m)})_{previous} \quad (6.29)$$

Finally, we backpropagate the error to the first hidden layer C_1. The local error at position (x,y) for a given map m from the layer C_1 is shown in the following equation :

$$e^{(1,m)}(x,y) = \delta^{(2,m)}(x/2, y/2) \times w^{(2)}_{m,m} \quad (6.30)$$

The local gradient is computed by multiplying the local error by the derivative of the activation function:

$$\delta^{(1,m)}(x,y) = \phi'(A^{(1,m)}(x,y)) \times \delta^{(2,m)}(x/2, y/2) \times w^{(2)}_{m,m} \tag{6.31}$$

Then, the update of the weight matrices $w^{(1)}_{ch,m}(u,v)$ where $(u,v) \in \{-2..2\}$ connecting the input map ch from to the map m from layer $C_1$is:

$$\begin{aligned}\Delta w^{(1)}_{ch,m}(u,v) &= -\eta \sum_{(x,y)=(0,0)}^{(44,44)} \left(\delta^{(1,m)}(x,y) e^{(ch)}(x+u+\tfrac{M_1}{2}, y+v+\tfrac{M_1}{2})\right) \\ &\quad +\alpha \Delta w^{(1)}_{ch,m}(u,v))_{previous}\end{aligned} \tag{6.32}$$

In a similar way, we compute the update of the bias:

$$\Delta b^{(1,m)} = -\eta \sum_{x,y} (\delta^{(1,m)}(x,y)) + \alpha(\Delta b^{(1,m)})_{previous} \tag{6.33}$$

6.2.4 The choice of parameters

As mentioned before, we choose to build $NC_1 = NS_2 = 6$ maps in layers C_1, and S_2; $NC_3 = NS_4 = 16$ maps in layers C_3, and S_4; 120 units in N_5; and $N_{Class} = 36$ units in layer N_6. The combination scheme between layer S_2 and layer C_3 is explained in the table 5.3 and shown in figure 6.3. The convolution window size $M_1 \times M_1 = M_3 \times M_3 = 5 \times 5$ for both convolution layers C_1 and C_3. The sub-sampling factor is two in each direction for both sub-sampling layers S_2 and S_4. We use linear activation functions in C_1 and C_3 and sigmoid activations functions in S_2, S_4, N_5 and N_6.

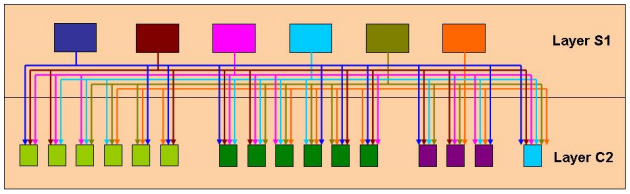


Figure 6.3: Feature Maps Combination Scheme

The different parameters governing the proposed architecture, i.e., the number of maps in the first layers, the number of neurons in the layer N_5, as well as the size of the receptive fields have been experimentally chosen. We expose below some of the most relevant experiments which justify our choices. All these experiments have been running over 100 iterations and for each one, we vary one parameter and we keep the best values for the remaining parameters. We choose to evaluate the performance through the

(M_1, M_3)	$(3,3)$	$(5,5)$	$(5,3)$	$(7,3)$	$(7,3)$
Recognition rate on ICDAR 2003 dataset	80.22%	84.53%	80.37%	81.23%	82.52%

Table 6.1: Performance as function of the convolution windowsize

N_{C1}	4	6	8	10
Recognition rate on ICDAR 2003 dataset	76.50%	84.53%	80.37%	80.80%

Table 6.2: Performance as function of the number of maps at layer C_1

character recognition rate on the ICDAR 2003 public set.

We have two convolution window sizes to choose, M_1 and M_3 corresponding to layers C_1 and C_3 respectively. Given that, the size of the maps at layer C_3 is smaller to the size of layer C_1, it is logic to choose M_3 smaller than or equal to M_1. A part from that, the size of the convolution mask should give a trade off between the precision of the feature extracted and the robustness against noise and distortion. Therefore, the convolution window sizes considered here are: $(M_1, M_3) = \{(3,3); (5,5); (5,3); (7,3); (7,5)\}$, Table 6.1 shows the performance of the system as function of the size of the convolution windows M_1 and M_3. The best result is given by $(M_1, M_3) = (5,5)$.

The second study concerns N_{C1}, the number of maps in the layer C_1. We do not study the number of maps in the layer C_3 because it is determined according to N_{C3} and the combination scheme used.
If the number of maps is small, the system does not learn well and the performance is bad, whereas, if the number of maps is very big, the system may over learn and the performance is also bad. Table 6.2 shows the recognition rates as function of N_{C1}. The best result is given by $N_{C1} = 6$. The last study concerns the number of units in layer N_5. This is the layer which requires the more weights. Similarly to the previous study, if the number of units is small, the system does not learn well and the performance is bad, whereas, if the number of maps is very big, the system may over learn and the performance is also bad. If the number of maps is small, the system does not learn well and performance is bad, whereas, if the number of units is very big, the system may over learn and the performance is also bad. Table 6.3 shows the recognition rates as function of the number of units in layer N_5. The best result is given by $N_5 = 120$.

Number of units in N_5	40	80	120	140
Recognition rate on ICDAR 2003 dataset	71.34%	76.21%	84.53%	79.08%

Table 6.3: Performance as function of the number of units at layer N_5

6.2.5 Testing Phase

After training, the system is ready to recognize automatically and rapidly color character images, without the need of any preprocessing steps as follows:

1. Resize the character image to the retina size (The size of the training input images are called retina size). Even though the aspect ratio may be not respected, the generalization ability of the system allows good recognition;

2. Feed the network with the three color channels of the resized character image as input;

3. Retrieve the output vector of the network $\{O_h\}_{h=1..N_{Class}}$;

4. The index corresponding to the highest component of this vector is considered as the recognized class.

6.3 Experimental results

To assess the performance of the proposed method, we use the sample subset of the ICDAR 2003 character database.

Table 6.4: Classification of images in ICDAR 2003 robust OCR Sample dataset.

Group	Number of Images
Clear	199
Background design	130
Multi-color character	54
Nonuniform lightning	40
Little contrast	37
Blurring	210
Serious distortion	28
Total	698

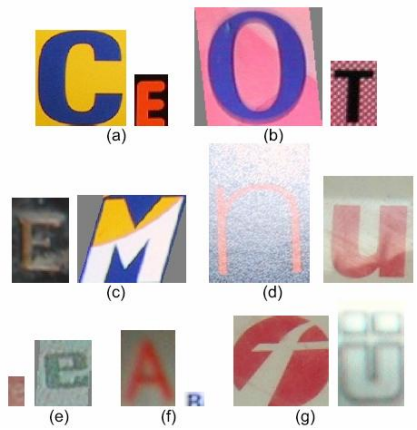


Figure 6.4: Examples of images in robust OCR Sample dataset classified into seven groups. (a) Clear. (b) Background design. (c) Multi-color character. (d) Nonuniform lightning. (e) Little contrast. (f) Blurring. (g) Shape distortion.

Given the wide range of variability contained in this database, and in order to compare our system to the recent works of Yokobayashi et al [YW05, YW06], we consider the classification proposed in [YW05, YW06] of 698 selected images from the above mentioned dataset, into seven groups according to the degree of image degradations and/or background complexity. Table 6.4 shows the number of images in each group and figure 6.4 shows examples of images in each of the seven groups.

Once the training has been performed, the system is now ready to be used. We present the image to be recognized to the network after having resized it to the system retina size. Then we take the highest output of the network last layer and we consider the corresponding class as the recognized character class.

Processing the entire test set (698 images) takes about 26 seconds, which means that an image is processed in approximately three-hundredth of a second.

Figure 6.5 shows the results of our method compared to [YW05] and [YW06]. The performance of our system reaches a recognition rate of 84.53% which outperforms the methods [YW05] and [YW06]: it ranges from 67.86% for seriously distorted images to 93.47% for clear images. Compared to the methods [YW05] and [YW06], the performance of our system is less affected by the categorization of the test set, especially in the case of non-uniform lighting condition and serious distortion, which is due to the good generalization ability of convolutional neural networks.

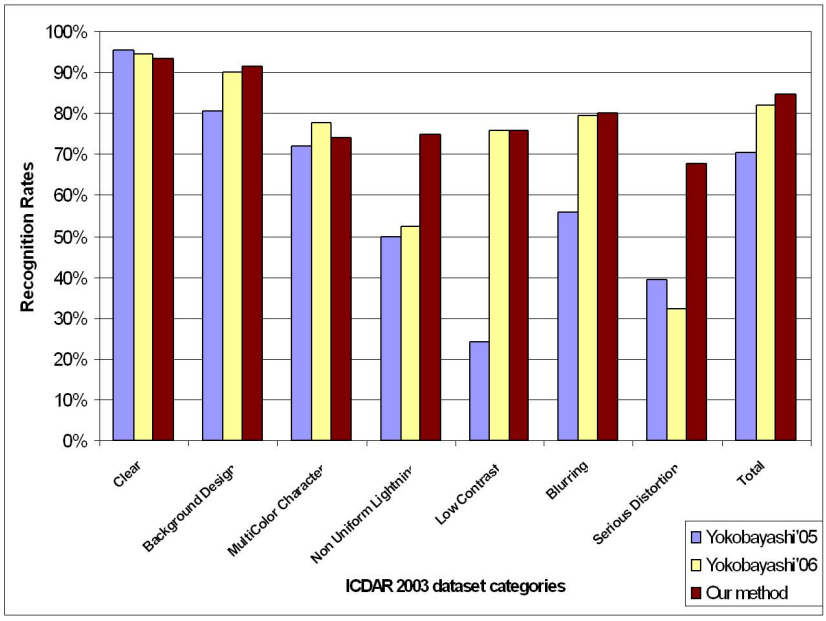


Figure 6.5: Recognition rates for each group of the test set

Figure 6.6 shows the top N cumulative recognition rates, where the correct answer is within the N best answers (i.e. the N highest outputs).

Here again our system outperforms the methods [YW05] and [YW06]. Furthermore, we notice that the cumulative recognition rate of the first two candidates is above 90%, showing the efficiency and the robustness of the proposed neural system.

6.4 Conclusion

In this chapter, we have proposed an automatic recognition system for complex color scene text image recognition, based on specific convolution neural network architecture. Thanks to supervised learning, our system does not require any tunable parameter and takes into account both color distributions and the geometrical properties of characters.

Only two state of the art methods have been tested on the public and actually challenging ICDAR 2003 robust reading data set. In this paper, we contribute to the state-of-the-art comparison efforts initiated in ICDAR 2003, and we show that our system outperforms these existing methods.

As a next step, we focus on building a global system which takes advantage from the text image segmentation module and the recognition modules.

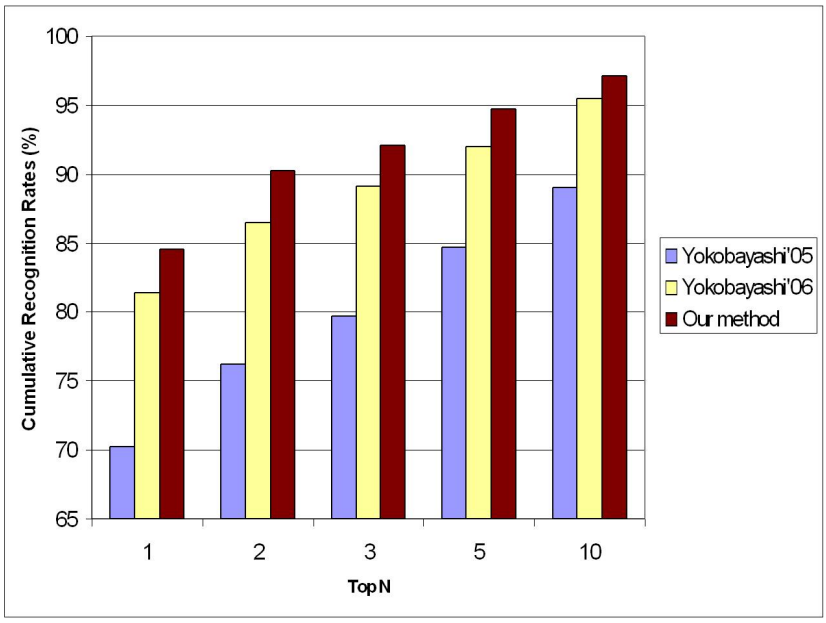


Figure 6.6: Cumulative recognition rates

Chapter 7

Discussion on the binarization step

On fait la science avec des faits,
comme on fait une maison avec des pierres.
Mais une accumulation de faits
n'est pas plus une science
qu'un tas de pierres n'est une maison.

Science is built up of facts,
as a house is with stones.
But a collection of facts
is no more a science
than a heap of stones is a house.
[Henri Poincaré] (1854--1912)

7.1 Introduction

We have presented two different systems for video text recognition. The first one is based on a combination of a binarization system and a printed document OCR, while the second one is based on a combination of a color text images segmentation system and a character recognition system.

At this stage, a question which rises spontaneously is *'is the binarization step in video text recognition opportune for machine learning based systems?'*. In fact, on one hand, a lot of the state of the art works are trying to take advantage from the classical OCR methods. Thus, they are claiming that the binarization step is a required step for video text recognition.

On the other hand, over the last two decades, the image text recognition community has shown an increasing interest in using gray and color images.

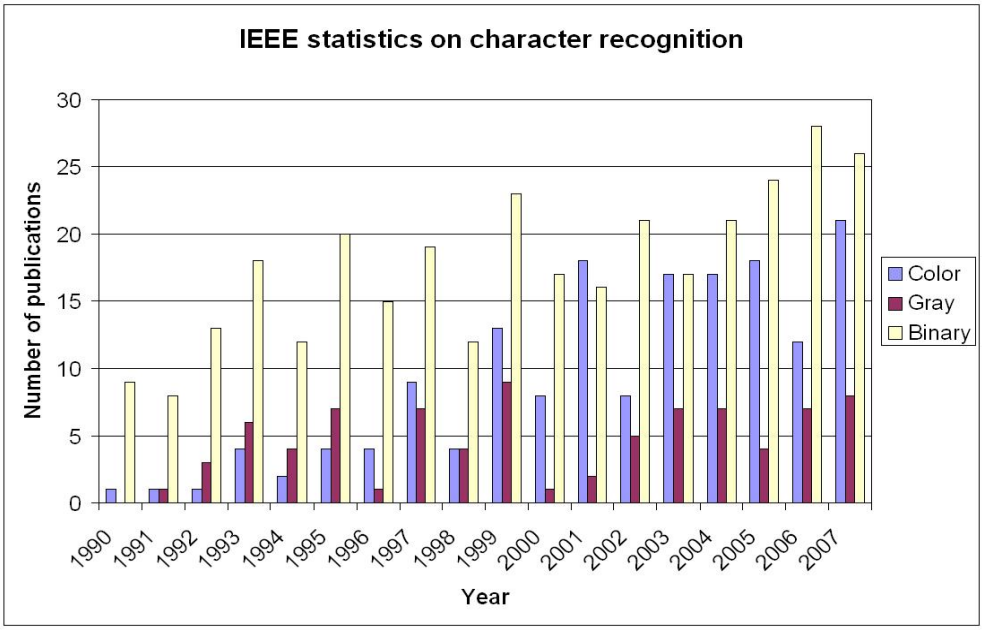


Figure 7.1: IEEE statistics on character recognition based on binary, gray and color images.

Figure 7.1 reports the number of IEEE publications that appeared from 1990 until 2007 that deal with character recognition. It represents three categories: character recognition based on binary images, on gray level images and on color images. We notice that, even though the number of works based on binary images is still the largest, the number of works based on gray or color images is increasing.

In video text recognition, studies on the relevance of the binarization step are still lacking. This chapter aims at completing the benchmark of this field by achieving some experiments to determinate what is the actual relevance of the binarization step in video text recognition.

In our experiments, we choose to use color images rather than gray level images, because the color cue could be more discriminative to distinguish the text in case of complex background, especially when the luminance values of some pixels are equal while their chrominance values are different.

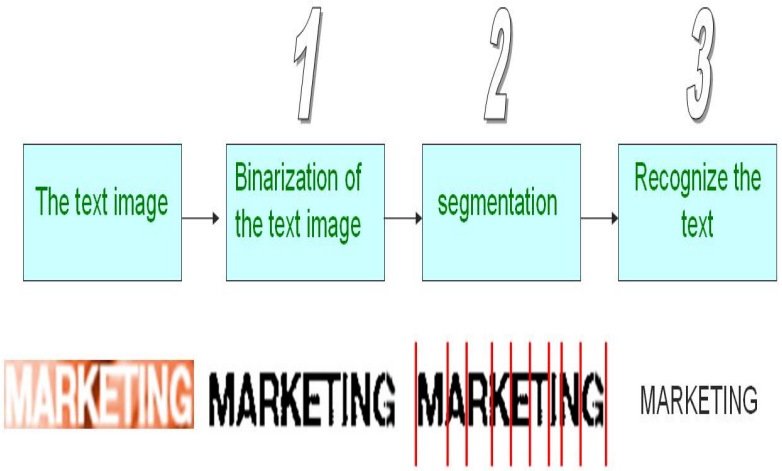


Figure 7.2: System A.

7.2 Experiments and Discussion

7.2.1 Video text recognition schemes

We propose to evaluate the importance of binarization using the modules presented in the previous chapters.

We propose to study two schemes:

- System A: a system where we apply the binarization module, then the segmentation module and the recognition module.
- System B: a system where we apply firstly the segmentation module and then the recognition module.

In the system A as illustrated in figure 7.2, we apply the binarization module, then the segmentation module and finally the recognition module. Therefore, the segmentation and the recognition systems are trained on binary images. This binary training set is obtained through the binarization of synthetic text images by the binarization module.

In the system B as illustrated in figure 7.3, we apply the segmentation module and then the recognition module. Therefore, the segmentation and the recognition systems are trained directly on color text images.

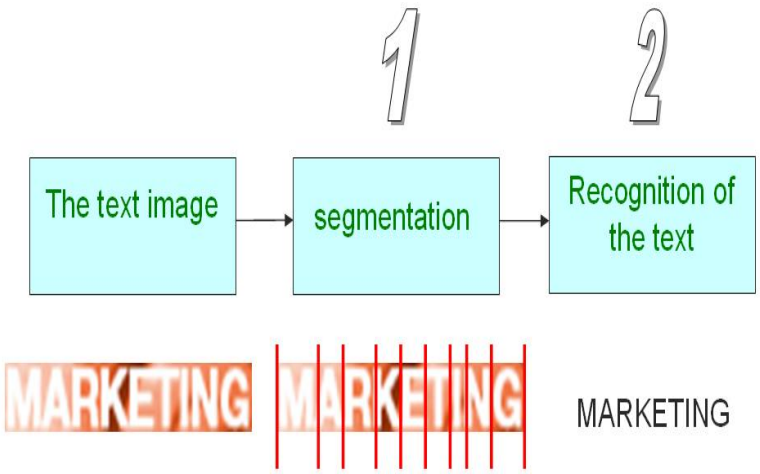


Figure 7.3: System B.

7.2.2 Data sets

We choose to use synthetic data for the training phase for both schemes in order to avoid hard and tremendous annotation task and because we are convinced of the good generalization ability of convolutional neural networks. As explained in the previous chapters, to build a synthetic image, we randomly choose a text color and a background color, we apply different types of noise (uniform and Gaussian noises), and then we apply smoothing filters in order to obtain images looking like real ones and presenting variability. Figure 7.4 shows some image examples from the synthetic dataset and the real dataset.

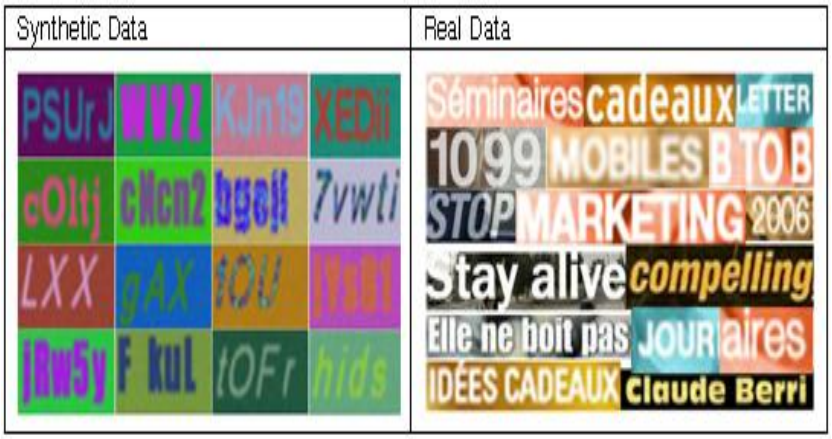


Figure 7.4: Datasets Examples.

The experimental results presented here are based on three sorts of datasets:

1. The training set containing 5000 synthetic text images, used to train the segmentation system and the recognition system.
2. The synthetic test set containing 5000 synthetic text images different from the training set.
3. The real test set containing 52 images of 469 characters collected from video frames and web pages. These images are considered as difficult images with complex background, bad resolution and a high level of variation according to luminance and text chrominance.

Figures 7.5 and 7.6 show some real text images processed respectively with system B and with system A.

Figure 7.5: Real text images processed with system B

Figure 7.6: Real text images processed with system A

7.2.3 Experiments on Segmentation

As explained before, the two first datasets are completely annotated, which means that the exact frontiers between characters and the text written in the images are known. Therefore the recognition and segmentation rates can be computed directly by comparing the results with the ground truth information. However, the real test set images are not annotated. Therefore, manual comparison of results is required. For the segmentation, we consider that a character is well segmented if 95% of its pixels is in between the frontiers detected by the system.

Figure 7.7 presents the Relative Operating Characteristic (ROC) curve for the segmentation system based on binary images and the ROC curve of the segmentation system based on color images. Given that the segmentation system can be also considered as a detection system of frontiers between characters, it is clearly relevant to show the ROC curves.

In these ROC curves, we represent the segmentation rate as a function of the threshold of the last layer in the segmentation system. In fact, the last layer C3 is composed of neurons with values ranging from -1 to +1. The system detects a frontier position when the neuron corresponding to this position is active, which means that the output of that neuron is above a predefined threshold. In our case, we report for each threshold the recall and the precision of the segmentation system. Thus, we can evaluate the best threshold, which corresponds to the best couple of precision and recall, to be used for each system when evaluating its performance on test data. The formula of the recall rate and the precision rate are reminded here:

$$Recall = \frac{Number\ of\ correctly\ identified\ frontier\ positions}{Number\ of\ desired\ frontier\ positions} \tag{7.1}$$

$$Precision = \frac{Number\ of\ correctly\ identified\ frontier\ positions}{Number\ of\ obtained\ frontier\ positions} \tag{7.2}$$

Both ROC curves concern the training set of each system. We notice that the result of the segmentation based on binary images is slightly better than the ones based on color images.

The table 7.1 and the table 7.2 present the results of both segmentation systems on the synthetic and the real test datasets. We notice that the results related to binary images are better than those related to color images on the synthetic sets. Whereas on the real test set, results on color images are slightly better than results on binary images.

The first conclusion we can draw according to the segmentation systems results is that there is not a huge difference between processing color images and binary images. This is probably due to the presence of many characters

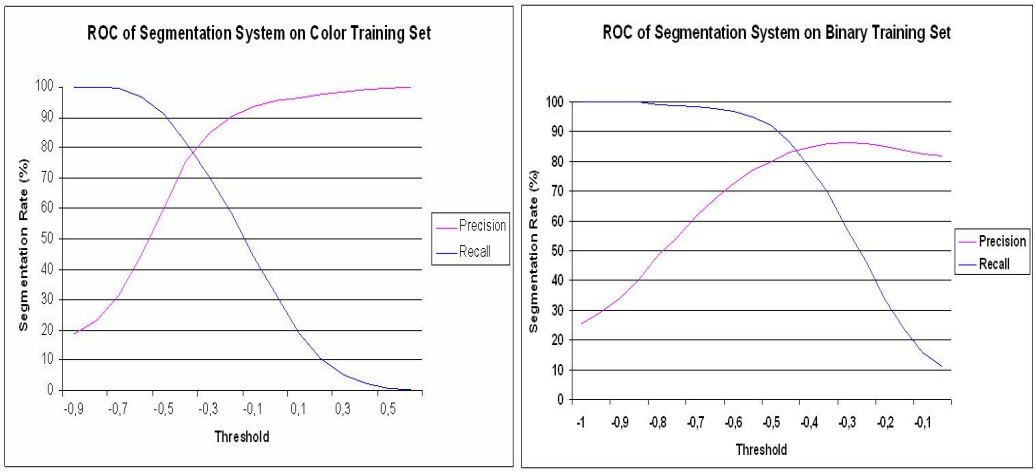


Figure 7.7: ROC curves of the segmentation system

Segmentation System	Binary Images		Color Images	
	Recall	Precision	Recall	Precision
Synthetic Test Set	86.72%	84.23%	86.18%	68.7%

Table 7.1: Performance of the segmentation system

Segmentation System	Binary Images	Color Images
Real Test Set	88%	89.1%

Table 7.2: Performance of the segmentation system on real test set.

Recognition System	Binary Images	Color Images
Training Set	99.53%	93.11%
Synthetic Test Set	86.66%	84.52%
Real Test Set	36.73%	81.90%

Table 7.3: Performance of the recognition system.

in the same image which gives a good ratio of variability and leads to good generalization in both cases.

7.2.4 Experiments on Recognition

Let us now focus on the recognition systems. The table 7.3 shows the results of the recognition system on the three datasets presented previously.

We notice that results related to binary images are better than those related to color images on the synthetic sets. Whereas on the real test set, results on color images are by far better than results on binary images.

To make sure that this decrease in the recognition system performance is not due to bad binarization, we counted semi-automatically the number of text pixels and the number of noisy pixels in each binarized image. We find that noisy pixels represent only 1.6% of the total number of black pixels in the whole real test dataset. This means that the binarization process is producing quite clean images.

7.2.5 Deductions

The deductions we can draw from these experiments are as follows:

- The binarization step is not required in machine learning methods. The recognition system based on CNNs gives good results while processing directly color images.
- The recognition system trained on color images performs generalization from synthetic data to real data. This is very interesting because we can avoid the hard and time consuming annotation task.
- The generalization ability is more difficult when using synthetic binary images. In fact, the system focuses mainly on character shapes. And given that, there are a huge number of different fonts in real data, it is quite impossible to represent all of them in a training dataset.

It is obvious, that texture and shape are important cues for text recognition in images and video frames. Nevertheless, the color information could

be also of major interest as a discriminative cue when backgrounds are really complex, with textures and high luminance variation.

7.3 Conclusion

In this chapter, we have conducted several interesting experiments based on the systems previously studied. We have also explored the generalization ability of the system from synthetic data to real data. The results have been mostly based on the segmentation and recognition rates. We have concluded from these experiments that the binarization step is not required in the systems based on machine learning algorithms and that over fitting is likely to occur when using binary synthetic text images. These conclusions being a complement to the studies that we have already done in video text recognition based on machine learning, the question of the relevance of binarization still remains an open question for video text recognition based on other methods and even for other pattern recognition applications.

Chapter 8

The iTRG - image Text Recognition Graph

La science est une chose merveilleuse...
tant qu'il ne faut pas en vivre !

Science is a wonderful thing...
if one does not have to earn one's living at it!
`[Albert Einstein] (1879--1955)`

8.1 Introduction

In the previous chapters, we presented two different systems:

1. The CTB binarization system combined with printed document OCR.
2. The CTS segmentation system combined with the CNNs based character recognition system.

At this stage, we are more concerned about the second system because we are interested in designing a specific complete system for video text recognition which does not need any use of classical OCRs.

In this chapter, we propose a new optimized text image recognition scheme through what we dubbed the iTRG - image Text Recognition Graph. This is a weighted directed acyclic graph $G = (V, E, W)$, which consists of a set of vertices (or nodes) V , a set of edges (or arcs) E and a set of edge weights W. In the iTRG, the edges are directed which means that a given edge $e(i, j)$ is oriented from i to j and that the edge $e(j, i)$ does not exist. The edges of the iTRG are also weighted, which means that a weight is assigned to each edge. In our case, this weight represents the probability of existence

of its corresponding edge as we will see later on. Finally, the iTRG is an acyclic graph which means that there is no cycles or loops. The proposed scheme consists of building the iTRG, then searching the best path which gives the sequence of characters contained in the processed image.

The remainder of this chapter is organized as follows. Section 8.2 describes in detail the different modules of the proposed scheme, especially the graph design. Experimental results are reported in Section 8.3. Conclusions are drawn in Section 8.4.

8.2 The construction of the iTRG

The construction of the iTRG scheme is based on five major modules:

1. The over segmentation module based on the segmentation system presented in chapter 5 with a specific post processing.

2. The graph connections builder module which takes the results of the over segmentation and build the connections between the graph vertices.

3. The recognition module based on the recognition system presented in chapter 6.

4. The graph weights calculator module based on both results of the over segmentation and the recognition to compute the weights of all the edges in the graph.

5. Best path Search which aims to find the path which gives the correct sequence of characters.

The figure 8.1 shows the connection between these different steps.

8.2.1 Over Segmentation

The over segmentation step is based on the segmentation system explained in chapter 5. The differentiation point here concerns the processing of the last layer C_3. In fact, in the segmentation process we consider only the neurons of the layer C_3 which are active, as corresponding to frontier positions between consecutive characters in the text image. In the over segmentation process, we consider also some other neurons which are not active, but which verify a set of predefined criteria. Thus, the output of the over segmentation process

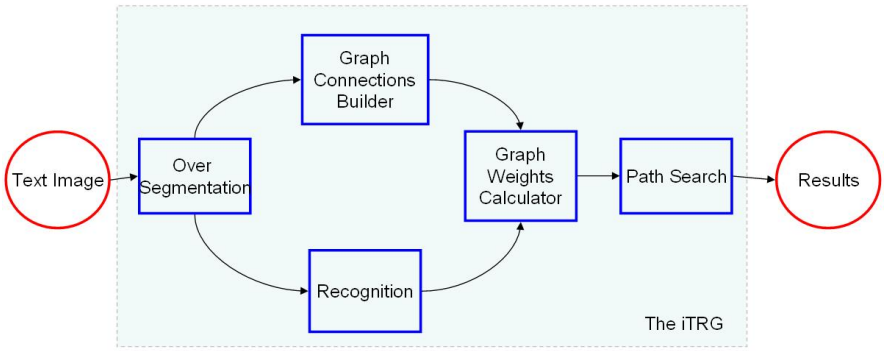


Figure 8.1: A simplified representation of the iTRG

are many possible frontier positions. The more relevant will be determined by the end of the overall process as detailed in next paragraphs.

Figure 8.2 shows the difference between the classical segmentation process and the over segmentation. The predefined criteria here are:

- the neuron output must be bigger than a chosen threshold. This threshold should be small enough in order not to miss any correct character frontier position.
- the neuron output must be a local maximum

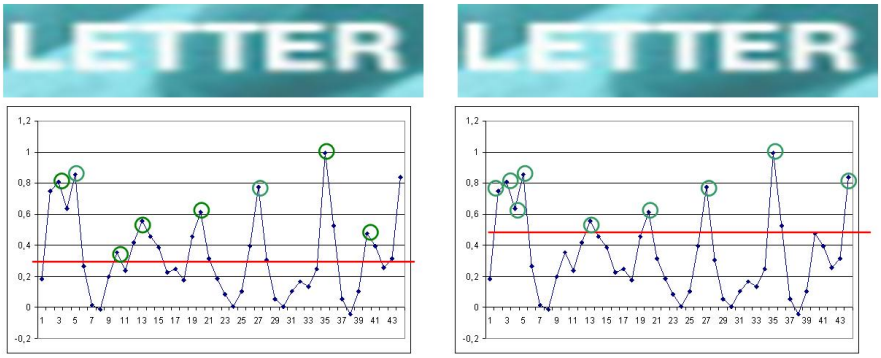
Moreover, we have to mention that the segmentation vector are normalized to the interval $[0, 1]$ so that to give more importance to the most significant peaks.

8.2.2 Graph's connections Builder

This module takes the over segmentation module results as input. Thus, every possible frontier position (corresponding to a local maximum output above a chosen threshold) is considered as a vertex in the iTRG (image Text Recognition Graph). The aim of this module is to build the connections between vertices.

Instead of connecting all vertices to each other, we choose to put some constraints in order to optimize the computation without loosing efficiency. These constraints are:

- A vertex i is connected to a vertex j if the distance between them $dist_{(i,j)}$ is within a chosen interval $[Dist_{min}, Dist_{max}]$. This means that we have to evaluate the minimum and the maximum width of characters.



(a) Over Segmentation

(b) Classical Segmentation

Figure 8.2: Over segmentation versus classical segmentation

- A vertex i cannot be connected to a vertex j if there is a vertex k between them which output $Output(V_k)$ is above a chosen threshold $HighOut$. In other words, when a vertex has a high output, we are sure that it is a frontier position. Paths which do not include it will not be considered.

$$Connect\ V_i\ to\ V_j\ if \begin{cases} dist_{(i,j)}) \in [Dist_{min}, Dist_{max}] \\ for\ every\ k \in [i,j]; Output(V_k) \leq HighOut \end{cases}$$

8.2.3 Recognition

The recognition module is based on the character recognition system presented in chapter 6. This module takes as input vertices positions and connections of the iTRG. Every two connected vertices correspond to a possible character image. These images are cropped from the original one and given to the recognition system. The output of this system is a vector R, where a given i^{th} item $R[i]$ represents the probability that the character belongs

to class i. Therefore, we consider only the maximum output and its corresponding class. The results of this module are given as input to the graph weights calculator module.

8.2.4 Graph Weights Calculator

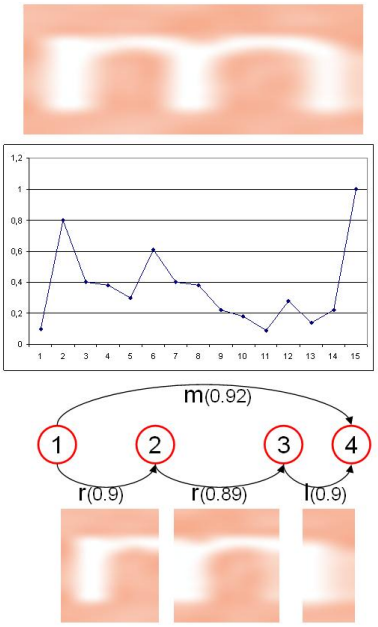
The graph weights calculator module takes as input both results of the over segmentation module and the recognition module to finish the construction of the iTRG. In fact, the weights of the edges connecting the vertices are computed as a function of the segmentation and the recognition outputs. We explore two equations to compute the edges weights. They are detailed below:

$$\begin{aligned} edgeweight_{i,j} = \\ & outRec_{i,j} \times \\ & [outSegm(nod_i) \times outSegm(nod_j)] \times \\ & [dist_{(i,j)}] \end{aligned}$$

$$\begin{aligned} edgeweight_{i,j} = \\ & outRec_{i,j} \times \\ & [outSegm(nod_i) \times outSegm(nod_j)] \end{aligned}$$

- $outRec_{i,j}$ represents the output of the recognition system
- $outSegm(nod_i)$ represents the output of the i_{th} neuron of the segmentation system. In fact, it is important to take into account the probability of being a frontier position for each vertex delimiting the image.
- $dist_{(i,j)}$ represents the distance in terms of number of units between the i_{th} neuron and the j_{th} neuron of the segmentation system.

Figure 8.3 shows the comparison between both equations. In fact, the last term in the first equation represents the width of the image segment. This factor is important because it normalizes somehow the edges' weights according to the image width. In fact, some characters could be easily recognized inside other characters, for example the character 'i' could be recognized in 'L', 'M', 'H', ...



Without taking into account the image width	Taking into account the image width
(1,2): 0.8x0.61x0.9=0.44 (2,3): 0.61x0.28x0.89=0.15 (3,4): 0.28x1x0.9=0.25 (1,2,3,4): 0.44+0.1+0.25=0.79	(1,2): 0.8x0.61x0.9x4=1.75 (2,3): 0.61x0.28x0.89x6=1.02 (3,4): 0.28x1x0.9x3=0.75 (1,2,3,4): 1.75+1.02+0.75=3.52
(1,4): 0.8x1x0.92=0.73	(1,4): 0.8x1x0.92x13=9.56
The best path is (1,2,3,4) ☹	The best path is (1,4) ☺

Figure 8.3: Importance of the image width in edge weights

8.2.5 Best Path Search

In graph theory, searching the best path is often related to the shortest path problem. This problem consists of finding a path between two vertices such that the sum of the weights of its constituent edges is minimized. When, there is only one source and one destination, this problem is called single-pair. The problem is also sometimes called:

- the single-pair shortest path problem, in which we have to find shortest paths from a source vertex v to a destination vertex v'
- The single-source shortest path problem, in which we have to find shortest paths from a source vertex v to all other vertices in the graph.
- The single-destination shortest path problem, in which we have to find shortest paths from all vertices in the graph to a single destination vertex v. This can be reduced to the single-source shortest path problem by reversing the edges in the graph.
- The all-pairs shortest path problem, in which we have to find shortest paths between every pair of vertices v, v' in the graph.

In our case, we consider that the problem is single pair. Even though, there could be M source vertices and M' destination vertices, because the

frontier of the character does not necessarily begin with the frontier of the image and the same at the end of the image, M and M' are most of the time lower than 3. Therefore, it is more appropriate to run occasionally a single pair algorithm twice or three times than choosing a multi-pair algorithm.

The most used algorithms for solving this problem are:

- Dijkstra's algorithm solves the single-pair, single-source, and single-destination shortest path problems.
- Bellman-Ford algorithm solves single source problem. It is usually used only when there are negative edge weights.
- A* search algorithm solves for single pair shortest path using heuristics to try to speed up the search.
- Floyd-Warshall algorithm solves all pairs shortest paths.
- Johnson's algorithm solves all pair's shortest paths, and may be faster than Floyd-Warshall on sparse graphs.
- Perturbation theory finds (at worst) the locally shortest path.

We choose to explore Dijkstra algorithm because of its well known efficiency.

Dijkstra algorithm

Dijkstra's algorithm, conceived in 1959, is a graph search algorithm that solves the shortest path problem for a graph with non negative edge weights and that is still widely used because of its performance and good computational cost in such problems. The algorithm is detailed below through 6 steps:

1. Element Creation: a distance list per vertex (the distances will be computed as function of the weights), a previous vertex list, a visited list and a current vertex.
2. Initialization:
    - All the values in the distance list are set to infinity except the starting vertex which is set to zero.
    - All values in visited list are set to false.
    - All values in the previous list are set to a special value signifying that they are undefined, such as null.

- The source vertex is set as the current vertex.

3. Mark the current vertex as visited.
4. Update distance and previous lists based on those vertices which are connected to the current vertex.
5. Update the current vertex to the unvisited vertex with the lowest distance from the source vertex.
6. Repeat (from step 3) until all nodes are visited.

Figure 8.4 details the pseudo-code of the Dijsktra algorithm.

'$u :=$ *node in* V_{tmp} *with smallest* $dist[]$': searches for the vertex u in the vertex set V_{tmp} that has the least $dist[u]$ value. That vertex is then removed from the set of visited vertices V_{tmp}.

$dist_between(u, v)$: calculates the length between the two neighbor nodes u and v.

$dist_tmp$ is the length of the path from the source node to the neighbor node v if it were to go through u.

If this path is shorter than the current shortest path recorded for v, that current path is replaced with this path. The previous array is populated with a pointer to the next node on the source graph to get the shortest route to the source.

This is the last step of the text recognition scheme. Once, we have processed the input image with the convolutional neural network segmentation system, build the graph nodes, processed each image segment with the character recognition system, and computed the edges' weights, now we apply the dijkstra algorithm to retrieve the best sequence of edges. Labels of this sequence give directly the recognized text. Figure 8.5 illustrates with an example the whole recognition scheme using the iTRG.

8.3 Experiments

To evaluate the performance of the iTRG, we run a variety of experiments based on a public database in order to contribute to the clear understanding of the current state of the art. Since the iTRG is meant to word recognition, we use the ICDAR 2003 test word data set. These word images are of different size (26×12, 365×58, 1249×223,...), different fonts, different colors, and present different kinds of distortion. We have classified manually 901 ICDAR word images with a total of 5698 characters into seven groups according to the

```
function Dijkstra(Graph, source, destination):

    // Initializations //
    for each vertex v in Graph:
        dist[v] := infinity                 // Unknown distance function from source to v
        previous[v] := undefined            // Previous node in optimal path from source

    dist[source] := 0                       // Distance from source to source
    V_tmp := the set of all vertices in Graph

    // The main loop //
    while V_tmp is not empty:
        u := vertex in V_tmp with smallest dist[]
        remove u from V_tmp
        for each neighbor v of u:           // where v has not yet been removed from V_tmp
        dist_tmp:= dist[u] + dist_between(u, v)
        if dist_tmp < dist[v]
          dist[v] := dist_tmp
          previous[v] := u

return previous[]
```

Figure 8.4: Pseudo code of the Disjkstra algorithm

complexity of the background, the variation of luminance and chrominance and the level of distortion. The seven groups are:

- Clear: this group contains 179 clear images with 1001 characters.
- Non uniform lightning: this group contains 105 images with 576 characters. These images have luminance variation and most of them are scene images.
- Multi color characters: this group contains 55 clear images with 306 characters. This is the smallest group and it contains words with multi color characters.
- Special design: this group contains 215 images with 1087 characters. These images have a special font or a special background design.
- Blur: this group contains 189 images with 985 characters and which are blurred.
- Low contrast: this group contains 61 images with 321 characters. The text in these images has a low contrast with the background. Actually, the ICDAR dataset contains more than 61 images in this category, but we choose to remove some of them because even human cannot recognize them.

- Serious distortion: this group contains 97 seriously distorted images with 521 characters. Generally, the serious distortion is due to a combination of many of the previously cited distortion.

Figure 8.6 shows some classified examples from the ICDAR 2003 test word dataset.

We tested our system according to two major elements, which are

- the performance of the segmentation
- the performance of the recognition

For the segmentation, we choose to evaluate the precision and the recall rates. In fact, the segmentation problem is equivalent to a border detection problem. The recall represents the probability that a randomly selected correct border is detected by the system, while the precision represents the probability that a randomly selected detected border is correct. The formula of Recall and Precision are reminded here after.

$$Recall = \frac{Number\ of\ Correct\ Borders}{Number\ of\ Desired\ Borders} \tag{8.1}$$

$$Precision = \frac{Number\ of\ Correct\ Borders}{Number\ of\ Obtained\ Borders} \tag{8.2}$$

The ICDAR public dataset is fully annotated through XML files containing the word written in every image and the estimated border column between each couple of consecutive characters. Given that a border position between two consecutive characters is not unique, we allow a margin of n columns. In other words, if the desired frontier position is P and the system find a border position between $P - n$ and $P + n$, then this position is considered as correct. A margin of $n = 2$ seems to be acceptable.

Figures 8.7 and 8.8 show the precision and the recall rates of the iTRG when test on the different categories of the ICDAR 2003 public database. The precision ranges from 63.44% for seriously distorted images to 94.3% for clear images. The recall ranges from 68.97% for seriously distorted images to 92.88% for clear images. These values are compared to the result of the previously presented system that we will call here the classical scheme and which consists on applying the segmentation system then the recognition system, without the use of the iTRG. We notice that there is an overall enhancement of 2% in precision and 19% in recall. In fact the over segmentation increases the probability to get the correct border positions among the set of detected

borders which increase the recall, furthermore relying on the recognition outputs to choose the best borders increases the precision. This is especially true for images with low contrast where the enhancement reaches 6.32% in precision and 23.6% in recall, and for images with multi color characters where the enhancement reaches 4.66% in precision and 20.46% in recall.

Figure 8.9 shows the word recognition rates of the iTRG on the different categories of the ICDAR 2003 public set compared to those of the classical scheme. The iTRG recognition performance is by far better than the classical performance. There is an enhancement of 10% over the whole data set and by more than 15% for clear images.

The figure 8.10 reports some relevant confusion rates between the different classes of alphanumerics processed by the iTRG. We notice that a great amount of errors is due to typical confusion, such as letter G wich is confused by 9% with letter C and by another 9% with letter O, or letter L which is confused by 5% with letter I, or the letter V which is confused by 20% with letter Y, or the letter O which is confused with 7% with digit 0. All these confusions suggest that the system will perform better if it is completed with some natural language processing module.

8.4 Conclusion

In this chapter, we propose a robust text recognition system based on a specific directed acyclic graph. To build this graph, we use text segmentation and character recognition modules based on convolutional neural networks. We contribute to the state-of-the-art comparison efforts initiated in ICDAR 2003 by evaluating the performance of our method on the public ICDAR 2003 test word set. The results reported here are encouraging and future directions could be towards including a statistical language modeling module.

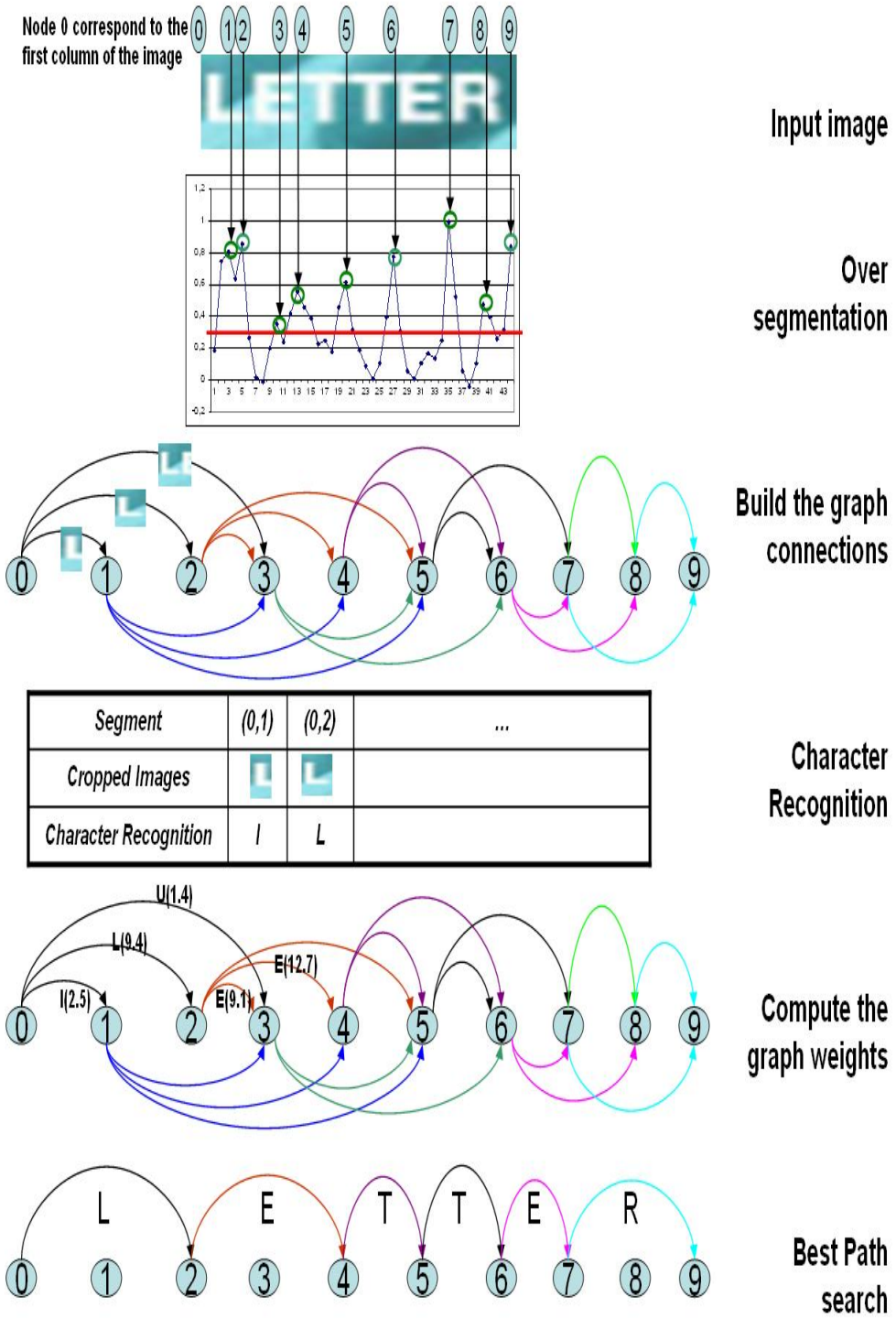


Figure 8.5: The iTRG

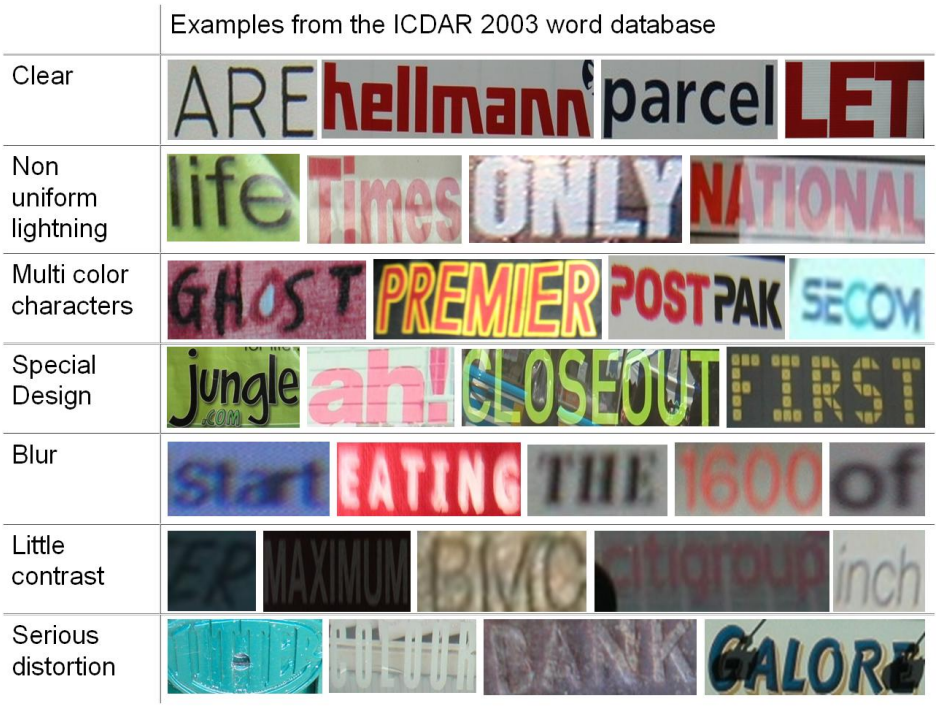


Figure 8.6: Examples from the ICDAR 2003 word dataset categories

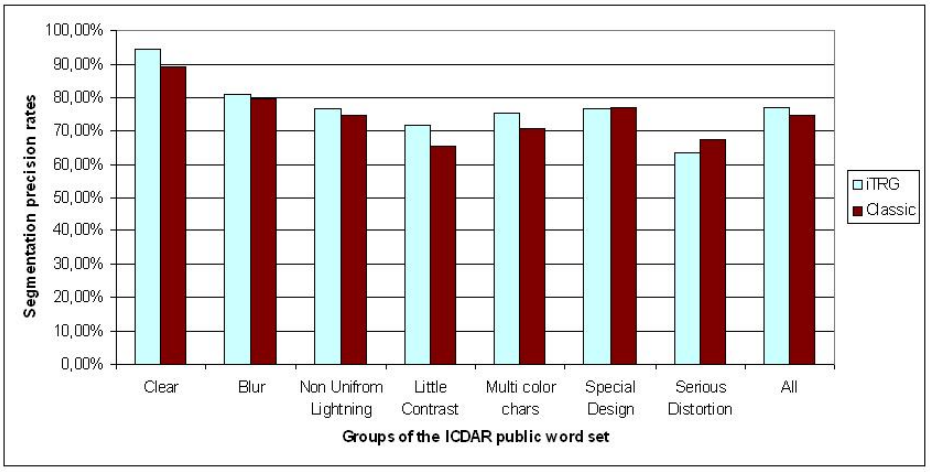


Figure 8.7: The precision of the segmentation in the iTRG and in the classical method tested on ICDAR 2003 public dataset

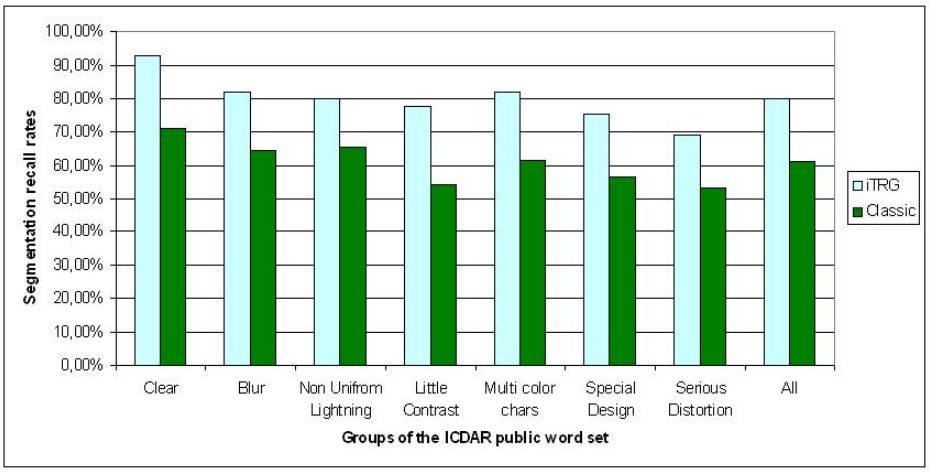


Figure 8.8: The recall of the segmentation in the iTRG and in the classical method tested on ICDAR 2003 public dataset

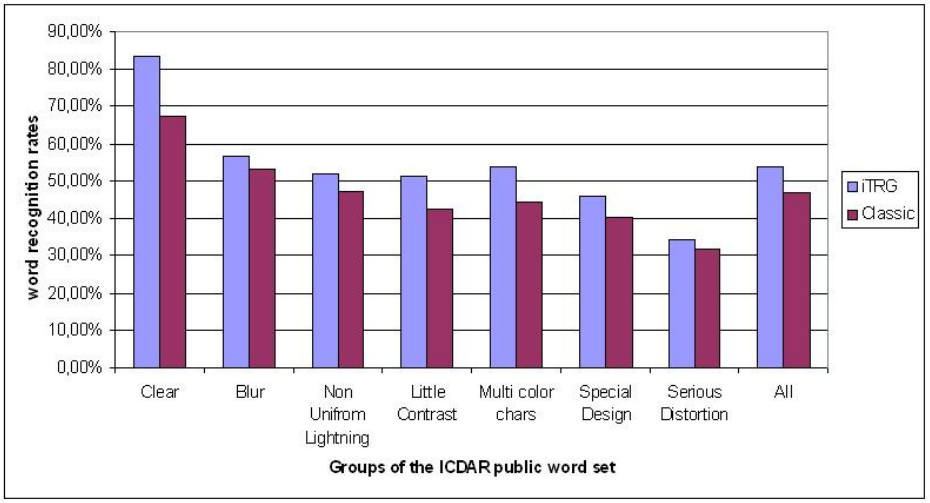
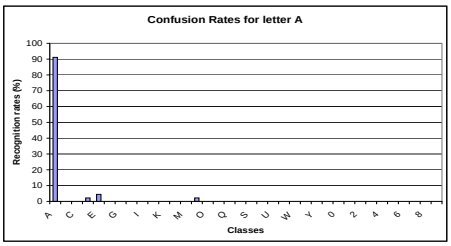
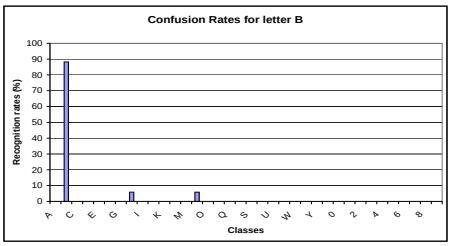
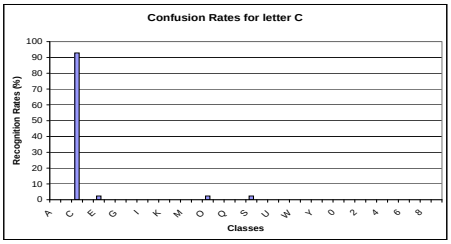
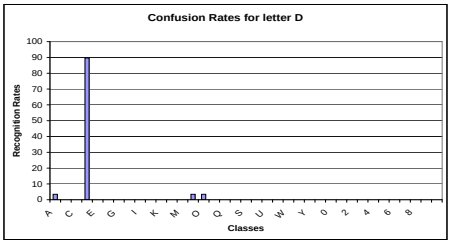


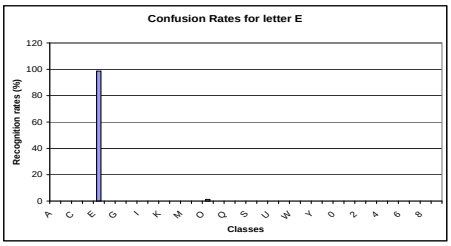
Figure 8.9: Recognition rates of the iTRG on ICDAR 2003 public dataset

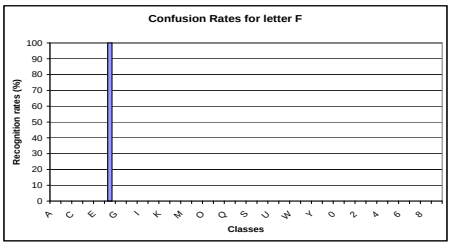

Confusion Rates for letter A
Recognition rates (%)
Classes

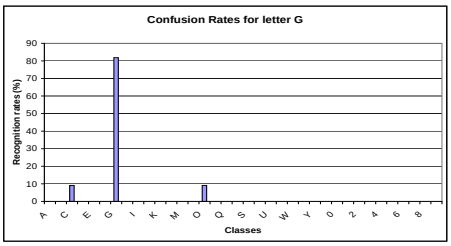

Confusion Rates for letter B
Recognition rates (%)
Classes

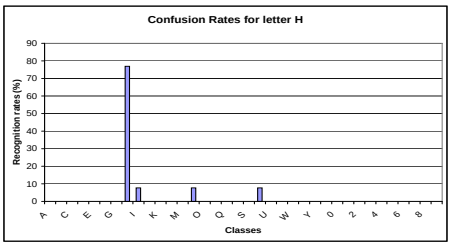

Confusion Rates for letter C
Recognition Rates (%)
Classes

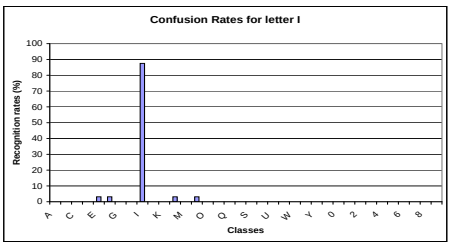

Confusion Rates for letter D
Recognition Rates
Classes

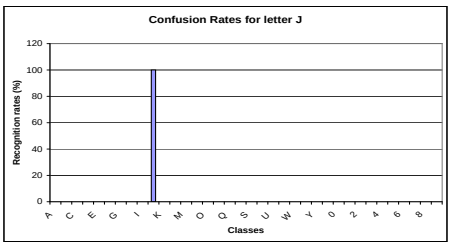

Confusion Rates for letter E
Recognition rates (%)
Classes

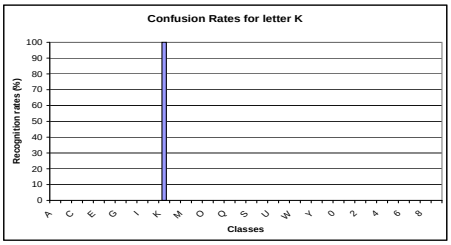

Confusion Rates for letter F
Recognition rates (%)
Classes

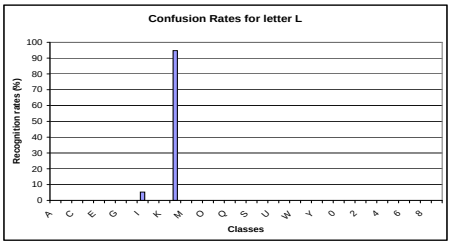

Confusion Rates for letter G
Recognition rates (%)
Classes

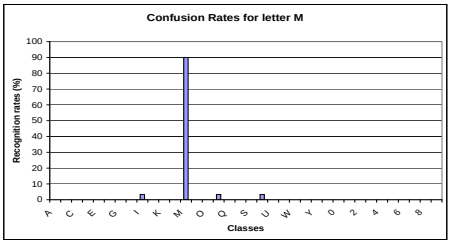

Confusion Rates for letter H
Recognition rates (%)
Classes

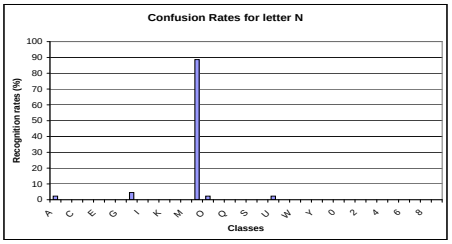

Confusion Rates for letter I
Recognition rates (%)
Classes

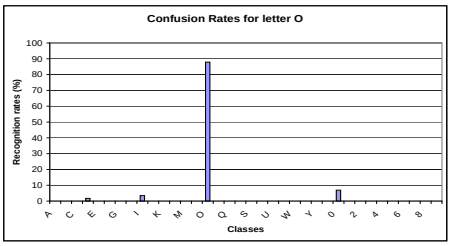

Confusion Rates for letter J
Recognition rates (%)
Classes

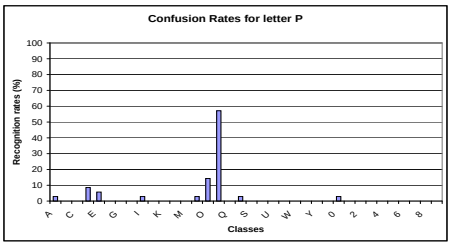

Confusion Rates for letter K
Recognition rates (%)
Classes

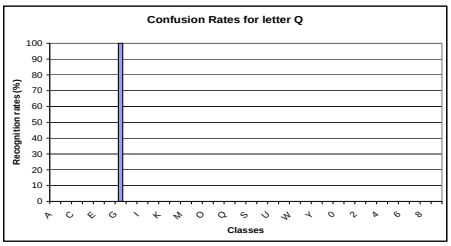

Confusion Rates for letter L
Recognition rates (%)
Classes

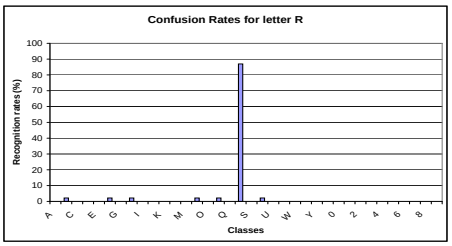

Confusion Rates for letter M
Recognition rates (%)
Classes

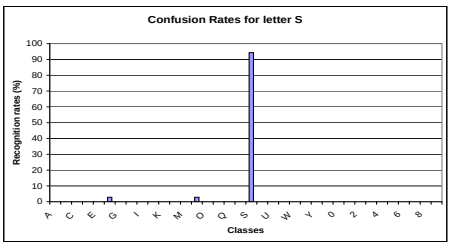

Confusion Rates for letter N
Recognition rates (%)
Classes

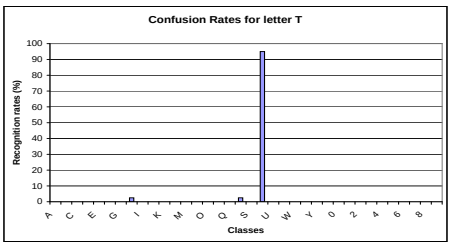

Confusion Rates for letter O
Recognition rates (%)
Classes


Confusion Rates for letter P
Recognition rates (%)
Classes


Confusion Rates for letter Q
Recognition rates (%)
Classes


Confusion Rates for letter R
Recognition rates (%)
Classes


Confusion Rates for letter S
Recognition rates (%)
Classes


Confusion Rates for letter T
Recognition rates (%)
Classes

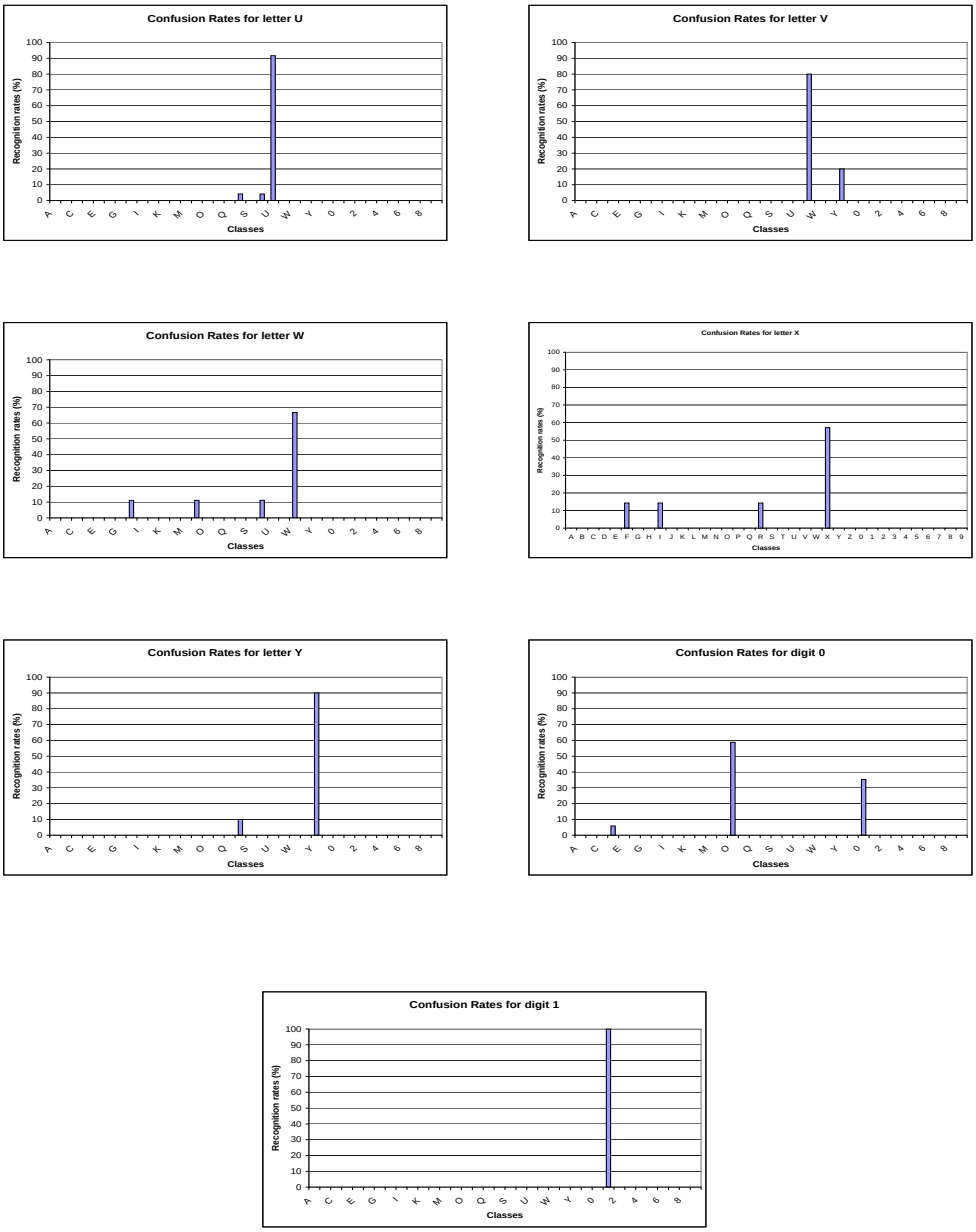


Figure 8.10: Most relevant Confusion Rates

Chapter 9

Conclusion and Perspectives

Quand les mots sont rares,
ils sont rarement dépensés en vain

When words are scarce,
they are seldom spent in vain
`[William Shakespeare] (1564--1616)`

In this thesis, we tackled the problem of video and image text recognition. The main purpose for building such a system is to fulfill the needs of growing content-based multimedia indexing, retrieval and management. Such systems are generally based on four main modules: detection, localization, extraction and recognition. We focus on the last module: recognition. This module can be broken into three steps, text image binarization, text image segmentation and character recognition.

We investigated each of the aforementioned steps and proposed novel methods based on a recent approach called Convolutional Neural Networks (CNNs). A Convolutional Neural Network is a connectionist model inspired by biological findings on the visual system of mammals. CNNs were initially used for handwritten digit recognition. Then, they have been successfully applied to many other visual pattern recognition problems [LBBH98], [GD04].

We proposed an automatic binarization method for color text areas in images or videos, which is robust to complex background, low resolution or video coding artifacts. The proposed system's topology is composed of four layers; the last two correspond to new concepts, namely, the upsampling and the inverse convolution. We call this system CTB, which stands for *Convolutional Text Binarizer*. It automatically learns how to perform binarization from a training set of synthesized text images and their corresponding desired binary images. We compared the proposed method with state-of-the-art

binarization techniques, with respect to Gaussian noise and contrast variations. Our experimental results highlited the robustness and the efficiency of our method. We conducted text recognition experiments on a database of images extracted from video frames and web pages. Two classical OCRs were applied on the CTB ouptut binary images. The results showed a strong enhancement of the recognition rate by more than 40%.

We also proposed an automatic segmentation system for characters in text color images. It is based on a new neuronal architecture with only three layers, insuring fast processing and robustness against noise, variations in illumination, complex background and low resolution. The main approach consists in an off-line training phase on a set of synthetic text color images, where the exact character positions are known. This operation allows adjusting the neural parameters and thus building an optimal non linear filter which extracts the best features in order to robustly detect the border positions between characters. The proposed method was tested on a set of synthetic text images to precisely evaluate its performance according to noise, and on a set of complex text images collected from video frames and web pages to evaluate its performance on real images. The results were encouraging with a good segmentation rate of 89.12%.

Then we proposed a character recognition system based on a classical architecture of CNNs with six layers. It automatically learns how to recognize characters without making any assumptions and without applying any preprocessing or post-processing. For this purpose, we used a training set of scene character images extracted from the ICDAR 2003 public database. The proposed method was compared to recent character recognition techniques in order to contribute to the state-of-the-art benchmark initiated in ICDAR 2003. Experimental results showed an average recognition rate of 84.53%, ranging from 93.47% for clear images to 67.86% for seriously distorted images which outperforms state of the art techniques.

Afterwards, we studied the relevance of the binarization step. Indeed, most works in video text recognition are assuming to start from binary images. Nevertheless an increasing interest is given to gray level and color images. We conducted some experiments on the methods we proposed until now. We re-trained the segmentation and the recognition system on binary images, binarized with our CTB and on non binary images. The results entail that the binarization step is not required in the systems based on machine learning algorithms and that the generalization ability to real data when learning from synthetic data is better with non binary text images.

Finally, we proposed a global scheme for color text recognition in images or videos, which is robust to complex background, low resolution or video coding artifacts. This scheme is based on a method that we call the image

Text Recognition Graph (iTRG) composed of five main modules: an image text segmentation module, a graph connection builder module, a character recognition module, a graph weight calculator module and an optimal path search module. The first two modules are based on segmentation and recognition systems described previously. We have been the first to evaluate our method on the public ICDAR 2003 word dataset. It achieved a word recognition rate of 83.72% on clear images.

Perspectives

Some possible enhancements can be investigated in the future.

- The letters confusion rates shown in the last chapter entail that these results can be reduced using statistical language modeling module. Such module aims at estimating the distribution of natural language as accurate as possible. Consequently, when the system confuses an 'O' with a 'Q' or an 'L' with an 'I', the linguistic model evaluates the probability of occurrence of the current processed character as function of its context (neighbor letters) and so, can correct the result. One can start with bigrams models.

- the segmentation and character recognition modules of the iTRG are trained separately. We believe that applying a co-training algorithm for these two modules can enhance the overall performance as the update of parameters will depend on both segmentation and recognition errors. The recent works of Collobert and Weston [CW08] which investigated this problem for natural language processing, can be a good starting point.

Bibliography

[Abb] http://www.abbyy.com.

[AD99] L. Agnihotri and N. Dimitrova. Text detection for video analysis. In *Proc. IEEE Workshop on Content-Based Access of Image and Video Libraries (CBAIVL '99)*, pages 109–113, 22 June 1999.

[Ber86] J. Bernsen. Dynamic thresholding of grey-level images. *ICPR'86*, pages 1251–1255, 1986.

[BGL93] J. Bertille, M. Gilloux, and M. Leroux. Recognition of handwritten words in a limited dynamic vocabulary. In *Proc. Third Int'l Workshop Frontiers in Handwriting Recognition*, pages 417–422, 1993.

[BLH93] Yoshua Bengio, Yann LeCun, and Donnie Henderson. Globally trained handwritten word recognizer using spatial representation, convolutional neural networks, and hidden markov models. In *NIPS*, pages 937–944, 1993.

[BMLC+92] C.J.C. Burges, O. Matan, Y. Le Cun, J.S. Denker, L.D. Jackel, C.E. Stenard, C.R. Nohl, and J.I. Ben. Shortest path segmentation: a method for training a neural network to recognize character strings. In O. Matan, editor, *Proc. International Joint Conference on Neural Networks IJCNN*, volume 3, pages 165–172 vol.3, 1992.

[CBD+89] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. In *Neural Computation*, volume 1, pages 541–551, 1989.

[CBD+90] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten digit

recognition with a back-propagation network. In *Advances in Neural Information Processing Systems*, pages 396–404. Morgan Kaufmann, 1990.

[CCC+04] Shyang-Lih Chang, Shyang-Lih Chang, Li-Shien Chen, Yun-Chung Chung, and Sei-Wan Chen. Automatic license plate recognition. *IEEE ITS*, 5(1):42–53, 2004.

[CdC93] C.H. Chen and J.L. de Curtins. Word recognition in a segmentation-free approach to ocr. In *ICDAR93*, pages 573–576, 1993.

[CGR04] Teo Boon Chen, D. Ghosh, and S. Ranganath. Video-text extraction and recognition. In *Proc. TENCON 2004. 2004 IEEE Region 10 Conference*, pages 319–322, Volume A,&1–24 Nov. 2004.

[CH97] Yuntao Cui and Qian Huang. Automatic license extraction from moving vehicles. *Image Processing, 1997. Proceedings., International Conference on*, 3:126–129 vol.3, Oct 1997.

[CHL05] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR (1)*, pages 539–546, 2005.

[CHLC05] Chien-Hsing Chou, Chih-Ching Huang, Wen-Hsiung Lin, and Fu Chang. Learning to binarize document images using a decision cascade. In *ICIP (2)*, pages 518–521, 2005.

[CL96] Richard G. Casey and Eric Lecolinet. A survey of methods and strategies in character segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(7):690–706, 1996.

[CLS+95] H.-P. Chan, S.-C. B. Lo, B. Sahiner, K. L. Lam, and M. A. Helvie. Computer-aided detection of mammographic microcalcifications: Pattem recognition with an artificial neural network. *Med. Phys.*, 22:1555–1567, 1995.

[CO03] Datong Chen and Jean-Marc Odobez. Sequential monte carlo video text segmentation. In *ICIP*, pages 21–24, 2003.

[CRV03] A. Calderón, S. Roa, and J. Victorino. Handwritten digit recognition using convolutional neural networks and gabor filter. In *Proceedings of the International Congress on Computational Intelligence*, 2003.

[CSB01] Datong Chen, K. Shearer, and H. Bourlard. Text enhancement with asymmetric filter for video ocr. In *Proc. 11th International Conference on Image Analysis and Processing*, pages 192–197, 26–28 Sept. 2001.

[CW08] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: deep neural networks with multitask learning. In *ICML*, pages 160–167, 2008.

[CYZW04] Xilin Chen, Jie Yang, Jing Zhang, and A. Waibel. Automatic detection and recognition of signs from natural scenes. *IEEE IP*, 13(1):87–99, Jan. 2004.

[DAS01] C. Dorai, H. Aradhye, and Jae-Chang Shim. End-to-end video-text recognition for multimedia content analysis. *Multimedia and Expo, 2001. ICME 2001. IEEE International Conference on*, pages 479–482, Aug. 2001.

[DG06] S. Duffner and C. Garcia. A neural scheme for robust detection of transparent logos in tv programs. In *In International Conference on Artificial Neural Networks (ICANN)*, volume 2, pages 14–23, sept. 2006.

[DG07] M. Delakis and C. Garcia. Text detection with convolutional neural networks. In *VISAPP*, volume 2, pages 14–23, 2007.

[DSS93] Harris Drucker, Robert E. Schapire, and Patrice Simard. Improving performance in neural networks using a boosting algorithm. In *Advances in Neural Information Processing Systems 5, [NIPS Conference]*, pages 42–49, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[Ear62] L.D. Earnest. Machine recognition of cursive writing. In *IFIP Congress*, 1962.

[EF71] P. Ekman and W. Friesen. Constants across culture in the face and emotion. *Journal of Personality and Social Psychology*, 17:124–129, 1971.

[Fas02] Beat Fasel. Multiscale facial expression recognition using convolutional neural networks. In *ICVGIP*, 2002.

[FMI83] K. Fukushima, S. Miyake, and T. Ito. Neocognitron: a neural network model for a mechanism of visual pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 13(5):826–834, 1983.

[FS98] E. Fukushima, K.and Kimura and H. Shouno. Neocognitron with improved bend-extractors. *IEEE World Congress on Computational Intelligence. The IEEE International Joint Conference on Neural Networks Proceedings, 1998.*, 2:1172–1175, May 1998.

[Fuk80] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Journal Biological Cybernetics*, 36(4):193–202, Apr 1980.

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

[Fuk01] K. Fukushima. Neocognitron of a new version: handwritten digit recognition. *Neural Networks, 2001. Proceedings. IJCNN '01. International Joint Conference on*, 2:1498–1503, 2001.

[GA00] C. Garcia and X. Apostolidis. Text detection and segmentation in complex color images. In *ICASSP*, volume 4, pages 2326–2329, June 2000.

[GAK98] U. Gargi, S. Antani, and R. Kasturi. Indexing text events in digital video databases. In *Proc. Fourteenth International Conference on Pattern Recognition*, volume 1, pages 916–918, 16–20 Aug. 1998.

[GD04] C. Garcia and M. Delakis. Convolutional face finder: a neural architecture for fast and robust face detection. *IEEE PAMI*, 26(11):1408–1423, 2004.

[GEF04] J. Gllavata, R. Ewerth, and B. Freisleben. Text detection in images based on unsupervised classification of high-frequency wavelet coefficients. In *ICPR04*, pages I: 425–428, 2004.

[Gep06] Alexander Gepperth. Visual object classification by sparse convolutional neural networks. In *ESANN*, pages 179–184, 2006.

[GESF04] Julinda Gllavata, Ralph Ewerth, Teuta Stefi, and Bernd Freisleben. Unsupervised text segmentation using color and wavelet features. In *CIVR*, pages 216–224, 2004.

[HDDK05] J. He, Q.D.M. Do, A.C. Downton, and J.H. Kim. A comparison of binarization methods for historical archive documents. In *ICDAR05*, pages I: 538–542, 2005.

[HK93] Tom Heskes and Bert Kappen. On-line learning processes in artificial neural networks. In J. Taylor, editor, *Mathematical Approaches to Neural Networks*, pages 199–233. Elsevier, Amsterdam, 1993.

[Hp] http://sourceforge.net/projects/tesseract-ocr.

[HP76] S.L. Horowitz and T. Pavlidis. Picture segmentation by a tree traversal algorithm. *ACM*, 2:368–388, 1976.

[HSAAEF06] Songtao Huang, M.A. Sid-Ahmed, M. Ahmadi, and I. El-Feghi. A binarization method for scanned documents based on hidden markov model. *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, page 4 pp, May 2006.

[HSW89] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[HW62] D. H. HUBEL and T. N. WIESEL. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *J Physiol*, 160:106–154, January 1962.

[HYZ02] Xian-Sheng Hua, Pei Yin, and Hong-Jiang Zhang. Efficient video text recognition using multiple frame integration. In *Proc. International Conference on Image Processing 2002*, volume 2, pages II–397–II–400, 22–25 Sept. 2002.

[iF89] Ken ichi Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks, 1989*, 2:183–192, 1989.

[JSVR05] C. Jacobs, P.Y. Simard, P. Viola, and J. Rinker. Text recognition of low-resolution document images. *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*, pages 695–699 Vol. 2, Aug.-1 Sept. 2005.

[KA04] D. Karatzas and A. Antonacopoulos. Text extraction from web images based on a split-and-merge segmentation method using colour perception. In *Proceedings of the 17th International Conference on Pattern Recognition. ICPR 2004*, volume 2, August 2004.

[KCC00] Sangshin Kwak, Yeongwoo Choi, and Kyusik Chung. Video caption image enhancement for an efficient character recognition. In *Proc. 15th International Conference on Pattern Recognition*, volume 2, pages 606–609, 3–7 Sept 2000.

[KHE05a] S. Kopf, T. Haenselmann, and W. Effelsberg. Robust character recognition in low-resolution images and videos. Technical Report TR-05-002, Department for Mathematics and Computer Science, University of Mannheim, 2005.

[KHE05b] S. Kopf, T. Haenselmann, and W. Effelsberg. Robust character recognition in low-resolution images and videos. Technical report, Department for Mathematics and Computer Science, University of Mannheim, April 2005.

[Kim04] In-Jung Kim. Multi-window binarization of camera image for document recognition. In *Proc. Ninth International Workshop on Frontiers in Handwriting Recognition IWFHR-9 2004*, pages 323–327, 26–29 Oct. 2004.

[KS04] E. Kavallieratou and S. Stamatatos. Discrimination of machine-printed from handwritten text using simple structural characteristics. In *Proc. 17th International Conference on Pattern Recognition ICPR 2004*, volume 1, pages 437–440, 23–26 Aug. 2004.

[LAKG98] M.J. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba. Coding facial expressions with gabor wavelets. In *Third IEEE International Conference on Automatic Face and Gesture Recognition*, pages 200–205, 14–16 Apr. 1998.

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

[LBOM98] Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural Networks: Tricks*

of the Trade, this book is an outgrowth of a 1996 NIPS workshop, pages 9–50, London, UK, 1998. Springer-Verlag.

[LDK00] Huiping Li, D. Doermann, and O. Kia. Automatic text detection and tracking in digital video. *IEEE IP*, 9(1):147–156, Jan. 2000.

[LE98] R. Lienhart and W. Effelsberg. Automatic text segmentation and text recognition for video indexing. Technical Report TR-98-009, Praktische Informatik IV, University of Mannheim, 1998.

[LFLM92] S. C. B. Lo, M. T. Freedman, J. S. Lin, and S. K. Mun. Computerassisted diagnosis of lung nodule detection using artificial convolution neural network. *in Proc. SPIE Med. Imag.: Image Processing*, 1898:859–869, 1992.

[LFLM93] S. C. B. Lo, M. T. Freedman, J. S. Lin, and S. K. Mun. Automatic lung nodule detection using profile matching and backpropagation neural network techniques. *J. Digital Imag.*, 6:48–54, 1993.

[Lie96] Rainer Lienhart. Automatic text recognition for video indexing. Technical Report TR-96-006, Praktische Informatik IV, University of Mannheim, 1996.

[LJB+95] Y. LeCun, L. D. Jackel, L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. A. Muller, E. Sackinger, P. Simard, and V. Vapnik. Learning algorithms for classification: A comparison on handwritten digit recognition. In *Neural Networks: The Statistical Mechanics Perspective*, pages 261–276, 1995.

[LL99] B. Q. Li and B. Li. Building pattern classifiers using convolutional neural networks. In *International Joint Conference on Neural Networks*, volume 5, pages 3081–3085, 1999.

[LLM06] Yuelong Li, Jinping Li, and Li Meng. Character recognition based on hierarchical rbf neural networks. *Intelligent Systems Design and Applications, 2006. ISDA '06. Sixth International Conference on*, 1:127–132, Oct. 2006.

[LPS+05] S.M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, R. Young, K. Ashida, H. Nagai, M. Okamoto, H. Yamaoto, H. Miyao, J. Zhu, W. Ou, C. Wolf, J.M. Jolion, L. Todoran,

M. Worring, and X. Lin. Icdar 2003 robust reading competitions: Entries, results and future directions. *International Journal on Document Analysis and Recognition*, 7(2–3):105–122, June 2005.

[LW02] R. Lienhart and A. Wernicke. Localizing and segmenting text in images and videos. *IEEE CASVT*, 12(4):256–268, 2002.

[MGS05] S. Marinai, M. Gori, and G. Soda. Artificial neural networks for document analysis and recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(1):23–35, Jan. 2005.

[MH01] T. Mita and O. Hori. Improvement of video text recognition by character selection. In *Proc. Sixth International Conference on Document Analysis and Recognition*, pages 1089–1093, 10–13 Sept. 2001.

[MMMK03] Masakazu Matsugu, Katsuhiko Mori, Yusuke Mitari, and Yuji Kaneda. Subject independent facial expression recognition with robust face detection using a convolutional neural network. *Neural Networks*, 16(5-6):555–559, 2003.

[MPBP91] J.V. Moreau, B. Plessis, O. Bougeois, and J.L. Plagnaud. A postal check reading system. In *ICDAR91*, pages 758–763, 1991.

[Nag92] G. Nagy. At the frontiers of ocr. *Proceedings of the IEEE*, 80(7):1093–1100, July 1992.

[Nag00] G. Nagy. Twenty years of document image analysis in pami. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(1):38–62, Jan 2000.

[Neu98] Claus Neubauer. Evaluation of convolutional neural networks. *IEEE Transactions on Neural Networks*, 9:685–696, jul. 1998.

[Nib86] W. Niblack. *An Introduction to Digital Image Processing*. N.J.: Prentice Hall, 1986.

[NP95] S. J. Nowlan and J. C. Platt. A convolutional neural network hand tracker. *Advances in Neural Information Processing Systems*, 7:901–908, 1995.

[NWF86] R. Nag, K.H. Wong, and F. Fallside. Script recognition using hidden markov models. In *ICASSP86*, pages 2071–2074, 1986.

[OCM07] Margarita Osadchy, Yann Le Cun, and Matthew L. Miller. Synergistic face detection and pose estimation with energy-based models. *J. Mach. Learn. Res.*, 8:1197–1215, 2007.

[Ots79] N. Otsu. A threshold selection method from gray-level histogram. *SMC-9*, 9:62–66, 1979.

[PSH+93] B. Plessis, A. Sicsu, L. Heute, E. Lecolinet, O. Debon, and J.V. Moreau. A multi-classifier strategy for the recognition of handwritten cursive words. In *ICDAR93*, pages 642–645, 1993.

[RBK96] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 203–208, 1996.

[Ros58] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Cornell Aeronautical Laboratory, Psychological Review*, 65(6):386–408, 1958.

[SCP+96] B. Sahiner, Heang-Ping Chan, N. Petrick, Datong Wei, M.A. Helvie, D.D. Adler, and M.M. Goodsitt. Classification of mass and normal breast tissue: a convolution neural network classifier with spatial domain and texture images. *Medical Imaging, IEEE Transactions on*, 15(5):598–610, Oct 1996.

[SG07a] Z. Saidane and C. Garcia. Automatic scene text recognition using a convolutional neural network. In *Proceedings of the Second International Workshop on Camera-Based Document Analysis and Recognition (CBDAR)*, sep 2007.

[SG07b] Z. Saidane and C. Garcia. Robust binarization for video text recognition. In *Proc. Ninth International Conference on Document Analysis and Recognition ICDAR 2007*, volume 2, pages 874–879, 23–26 Sept. 2007.

[SKHS98] T. Sato, T. Kanade, E.K. Hughes, and M.A. Smith. Video ocr for digital news archive. In *Proc. IEEE International Workshop on Content-Based Access of Image and Video Database*, pages 52–60, 3 Jan. 1998.

[SSHP97] J. Sauvola, T. Seppänen, S. Haapakoski, and M. Pietikäinen. Adaptive document binarization. In *In International Conference on Document Analysis and Recognition*, volume 1, 1997.

[SSP03] Patrice Y. Simard, Dave Steinkraus, and John C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *ICDAR '03: Proceedings of the Seventh International Conference on Document Analysis and Recognition*, page 958, Washington, DC, USA, 2003. IEEE Computer Society.

[TA03] S. Tekinalp and A.A. Alatan. Utilization of texture, contrast and color homogeneity for detecting and recognizing text from video frames. In *Proc. International Conference on Image Processing ICIP 2003*, volume 3, pages III–505–8, 14–17 Sept. 2003.

[TB03] F.H.C. Tivive and A. Bouzerdoum. A new class of convolutional neural networks (siconnets) and their application of face detection. *Neural Networks, 2003. Proceedings of the International Joint Conference on*, 3:2157–2162 vol.3, July 2003.

[TJCY07] Jia Tse, Christopher Jones, Dean Curtis, and Evangelos Yfantis. An ocr-independent character segmentation using shortest-path in grayscale document images. *icmla*, pages 142–147, 2007.

[WD02] Christian Wolf and David S. Doermann. Binarization of low quality text using a markov random field model. In *ICPR (3)*, pages 160–163, 2002.

[Wer90] P.J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, Oct. 1990.

[WKT01] T. Wakahara, Y. Kimura, and A. Tomono. Affine-invariant recognition of gray-scale characters using global affine transformation correlation. *IEEE Trans. Pattern Anal. Machine Intell.*, PAMI-23:384–395, August 2001.

[WMR99] V. Wu, R. Manmatha, and E.M. Riseman. Textfinder: an automatic system to detect and recognize text in images. *IEEE PAMI*, 21(11):1224–1229, Nov. 1999.

[YW05] M. Yokobayashi and T. Wakahara. Segmentation and recognition of characters in scene images using selective binarization in color space and gat correlation. *Eighth International Conference on Document Analysis and Recognition ICDAR'05*, 1:167–171, August 2005.

[YW06] M. Yokobayashi and T. Wakahara. Binarization and recognition of degraded characters using a maximum separability axis in color space and gat correlation. *18th International Conference on Pattern Recognition ICPR 2006*, 2:885–888, August 2006.

[YZHT03] Hao Yan, Yi Zhang, Zeng-Guang Hou, and Min Tan. Automatic text detection in video frames based on bootstrap artificial neural network and ced. In *WSCG*, 2003.

[ZC03] DongQuin Zhang and Shih-Fu Chang. A bayesian framework for fusing multiple word knowledge models in videotext recognition. *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, 2:II–528–II–533 vol.2, June 2003.

[ZKJ95] Yu Zhong, K. Karu, and A.K. Jain. Locating text in complex color images. In *Proc. Third International Conference on Document Analysis and Recognition*, volume 1, pages 146–149, 14–16 Aug. 1995.

List of Figures

List of Tables