Lecture automatique de tickets de caisse

Rizlène Raoui-Outach

Communauté
UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTE UNIVERSITE GRENOBLE ALPES

Spécialité : **Informatique**

Arrêté ministériel :

Présentée par

« Rizlène RAOUI-OUTACH »

Thèse dirigée par **Patrick LAMBERT, Professeur,** et
Co-encadrée par **Alexandre BENOIT, Maître de conférences,**

préparée au sein du **Laboratoire d'informatique, Systèmes, Traitement de l'Information et de la Connaissance (LISTIC)**
dans **l'École Doctorale Sciences et Ingénierie des Systèmes, de l'Environnement et des Organisations (SISEO)**

Lecture automatique de tickets de caisse.

Thèse soutenue publiquement le **« 19 Juillet 2019 »**,
devant le jury composé de :

M. Christophe DUCOTTET
Professeur, Université de Saint-Etienne, Président
Mme. Jenny BENOIS-PINEAU
Professeure, Université de Bordeaux, Rapportrice
M. Jean-Yves RAMEL
Professeur, Université de Tours, Rapporteur
M. Patrick LAMBERT
Professeur, Université Savoie Mont Blanc, Directeur de thèse
M. Alexandre BENOIT
Maître de Conférences, Université Savoie Mont Blanc, Co-Encadrant de thèse
M. René LE-CAIGNEC
PDG, AboutGoods Company, Membre

Remerciements

Cette thèse en convention CIFRE a été menée au sein du Laboratoire d'Informatique, Systèmes, Traitement de l'Information et de la Connaissance (LISTIC) de l'Universite de Savoie Mont Blanc, en collaboration avec l'entreprise AboutGoods Company à Annecy. J'adresse donc mes premiers remerciements à ces trois structures qui m'ont accueillie dans leurs locaux.

Je tiens à remercier Madame Jenny Benois-Pineau, Monsieur Jean-Yves Ramel pour avoir accepté de rapporter sur ma thèse ainsi que M. Christophe Ducottet. Les remarques constructives qui accompagneront leur rapport me seront d'un grand bénéfice.

Ce travail n'aurait pas pu être complet sans l'encadrement de Mr Patrick Lambert, Professeur à l'Université de Savoie Mont Blanc et Mr Alexandre Benoît, Maître de Conferences à l'Universite de Savoie Mont Blanc. Leurs conseils et leurs qualités humaines m'ont été indispensables tout au long de ce travail.

Je remercie pareillement, Mr René Le Caignec, Président de l'entreprise AboutGoods Company, de m'avoir fait confiance en me proposant de mener ce projet, crucial pour son entreprise. Je remercie également tous les collaborateurs de l'entreprise AboutGoods Company pour leur sympathie et en particulier Mme Cécile Million-Rousseau qui m'a accompagné tout au long de ce travail.

Je tiens à remercier tous les membres du LISTIC pour leur accueil, leurs encouragements et leur sympathie. Je remercie en particulier le personnel administratif, Joelle et Assia pour m'avoir aidé dans les formalités administratives tout le long de ces années.

C'est une période marquante de ma vie qui s'achève, qui m'a permis de faire la connaissance de nombreux autres doctorants au sein du LISTIC : Rihab, Meriem, Amina,Hela, Siwar, Quentin, Cedric, Matthias, Charles et Mathilde, autant de personnes avec qui j'ai partagé de très bons moments et que j'ai pu avoir des échanges très intéressants.

Enfin, je ne pourrais finir ces remerciements sans penser à ma famille pour leur soutien, notamment mon mari Jaouad dont l'affection et l'encouragement constants m'ont été d'un grand réconfort et ont contribué à l'aboutissement de ce travail. Je remercie également mon oncle Lahsen qui a toujours su m'apporter les bons conseils au bon moment. Enfin, je remercie affectueusement ma mère et mon père. Tout au long de mon cursus, ils m'ont toujours soutenu, encouragé et aidé. Une pensée à mes frères et soeurs, belles-soeurs et amis pour leurs soutiens et leur écoute.

Résumé

L'augmentation du volume de données multimédia et notamment la dématérialisation des documents papiers impose la mise en place de solutions permettant d'analyser automatiquement ces documents afin de faciliter leur stockage et leur exploitation.
Par ailleurs, il existe actuellement un fort intérêt des entreprises ou des instituts pour accéder aux informations de consommation des populations ou groupes de population afin d'avoir une meilleure compréhension du comportement des consommateurs. Le ticket de caisse est une solution permettant d'obtenir ces informations sans solliciter fortement le consommateur. L'objectif de cette thèse est donc de proposer une solution permettant d'analyser automatiquement le contenu d'un ticket de caisse à partir d'une photo prise par un smartphone.

Nous commençons par expliquer les objectifs industriels et, au travers du développement d'un démonstrateur, nous mettons en évidence les verrous scientifiques de la réalisation d'un tel système, de l'acquisition de l'image à l'extraction des données textuelles contenues dans le ticket. À l'issue de cette étude, nous proposons une chaîne de traitement originale pour répondre au mieux à toutes les attentes et contraintes.
Ensuite, nous réalisons un état de l'art détaillant les méthodes de détection d'objets basées notamment sur les réseaux de neurones profonds (détection de logo, détection de texte...). Nous présentons également les méthodes de reconnaissance de texte et les outils associés existants (OCR). Enfin nous terminons par évoquer quelques approches concernant l'analyse sémantique.
La première partie de la réalisation de la chaîne est la phase de pré-traitement qui va permettre de vérifier la présence d'un ticket dans l'image, de le localiser afin de le rogner et de le redresser, puis de déterminer l'enseigne de ticket. Dans le but de minimiser les fausses alarmes, chacun de ces objectifs est obtenu à l'issue de la fusion du résultat de deux méthodes basées sur des sources différentes (image et texte).
La deuxième partie consiste à analyser le contenu du ticket de caisse, en commençant par la segmentation sémantique des zones du ticket de caisse (en-tête, logo, liste de produits, bas de ticket, etc.), puis en réalisant la reconnaissance optique des zones de texte et enfin en appliquant une analyse sémantique afin d'extraire les différentes informations pertinentes.

mots-clés : Compréhension d'images de ticket de caisse, Réseaux de neurones convolutifs profonds, Détection d'objets, Analyse sémantique

Abstract

The large increase in multimedia data volume and especially the dematerialization of paper documents requires the implementation of solutions to automatically analyze these documents in order to facilitate their storage and their use. Moreover, there is currently a strong interest of companies or institutes to access consumer information of populations or population groups in order to have a better understanding of consumer behavior. The sales receipt is a solution to obtain this information without strongly soliciting the consumer. The objective of this thesis is to propose a solution to automatically analyze the contents of a sales receipt from a photo taken by a smartphone.

We begin by explaining the industrial objectives and, through the development of a demonstrator, we highlight the scientific obstacles of the realization of such a system, from the acquisition of the picture to the extraction of the textual data contained in the ticket. At the end of this study, we propose an original processing chain to best meet all expectations and constraints. Then, we realize a state of the art detailing methods of detection of objects based in particular on deep neural networks (logo detection, text detection...). We also present text recognition methods and existing associated tools (OCR). Finally, we end up evoking some approaches concerning semantic analysis. The first part of the realization of the chain is the pre-treatment. This phase has several goals : checking the presence of a sale receipt within the image, ticket in order to crop it and straighten it, and then to determining the brand of the receipt. In order to minimize false alarms, each of these objectives is obtained after merging the results of two methods based on different sources (image and text). The second part is to analyze the content of the receipt, starting with the semantic segmentation of the receipt areas (header, logo, product list, bottom of receipt, etc.), then performing optical recognition and finally applying a semantic analysis to extract the different relevant information

Keywords : Receipt Image Understanding, Deep Convolutional Neural Networks, Object Detection, Semantic Analysis

Table des matières

1. Introduction Générale

Dans ce chapitre, nous présentons le contexte applicatif des travaux de cette thèse. Dans un premier temps, nous introduisons les problèmes liés à l'analyse de données et plus particulièrement de documents dématérialisés. Ensuite, nous expliquons l'intérêt de notre travail dans le domaine spécifique de la lecture automatique de tickets de caisse. Enfin, nous donnons la structure de ce manuscrit.

1.1. Le Big Data et l'analyse de documents dématérialisés

Avec le développement des nouvelles technologies, d'internet et des réseaux sociaux, la production de données numériques est devenue phénoménale et sa croissance est constante. C'est ce que l'on appelle le « Big Data » . L'expression « Big Data » désigne la masse des données numériques produites par les entreprises et les particuliers dont les caractéristiques spécifiques sont la très grande Volumétrie, l'hétérogénéité ou Variété et la grande Vitesse d'évolution. Ce sont les classiques « 3V », auxquels on ajoute parfois deux autres « V », pour Véracité (fiabilité de la donnée), et Valeur (importance de la donnée). Tout cela requiert des outils informatiques spécifiques de plus en plus sophistiqués pour le stockage, le transfert, la visualisation et l'analyse. La très grande volumétrie et la variété sont peut-être ce qu'il y a de plus spectaculaire dans le Big Data : des trillions d'octets de données numériques sont générés quotidiennement, ces informations provenant de sources multiples et variées. Par exemple, dans le monde du multimédia on compte, chaque jour, plus de 50 millions de messages envoyés sur Twitter et 300 millions de photos échangées sur Facebook. Ce réseau social analyse environ 500 téraoctets de données par jour. Dans le domaine de la vidéo en ligne, YouTube récolte 400 heures de contenu chaque minute. Ces dernières années, cette explosion des données numériques a fortement mobilisé le monde de la recherche afin de trouver de nouvelles manières de stocker, mettre à disposition, transmettre et analyser ces données afin de répondre à de nouvelles demandes pouvant venir aussi bien des entreprises, des organismes publics que des individus.

Dans une démarche qui s'inscrit également dans le Big Data, on constate que de plus en plus de documents sont dématérialisés. Autrement dit, dans le but de faciliter leur stockage et leur exploitation, les documents papier sont remplacés par des documents numériques. La dématérialisation de documents touche de nombreux domaines et très peu d'entreprises peuvent y échapper. Deux types de dématérialisation coexistent aujourd'hui. Il y a d'une part la dématérialisation dite « native », c'est-à-dire que les documents sont uniquement disponibles numériquement et sont utilisés directement sous cette forme. Mais on utilise également la dématérialisation « duplicative » de documents

initialement disponibles au format papier puis numérisés. Dans ce dernier cas, on réalise une numérisation du document papier générant un fichier image (on parle de « scan ») dans le but de valoriser l'information et faciliter son traitement.

L'analyse automatique des données contenues dans ces fichiers images est alors quasiment indispensable si l'on veut une exploitation efficace de ces documents. C'est à ce type de dématérialisation que nous nous intéresserons dans ce travail.

Depuis longtemps, de très nombreux travaux se sont intéressés à l'analyse automatique de documents. On peut distinguer les documents manuscrits et les documents imprimés. La problématique de l'analyse de documents manuscrits est bien sûr plus complexe, même si des solutions commerciales existent maintenant et fonctionnent avec de bonnes performances dans des situations favorables. L'analyse de documents imprimés, c'est-à-dire des documents mêlant des graphiques et des textes imprimés tels que des factures, des cartes d'identité, des tickets de transport ou encore des tickets de caisse, constitue une problématique a priori beaucoup plus facile que celle des documents manuscrits. Elle peut néanmoins être rendue difficile si le texte est « enfoui », si le support papier ou la qualité d'impression sont dégradés et si l'acquisition n'est pas maîtrisée. Beaucoup de travaux existent dans ce domaine [1, 2, 3, 4], car les besoins et problématiques autour de la numérisation de documents sont nombreux.

De nombreuses applications pratiques découlent de la lecture automatique d'un document. Depuis longtemps déjà, des systèmes existent pour le tri du courrier [5] ou le traitement des chèques dans les banques [6]. La difficulté dans ces deux situations est essentiellement liée au fait que les documents concernés sont manuscrits. Mais même dans le cas de documents imprimés, la lecture automatique peut présenter un grand intérêt. Pour une entreprise, cela peut par exemple permettre un traitement plus efficace des factures qui arrivent encore souvent sous format papier [7, 8, 9] et la gestion des missions pour lesquelles les justificatifs sont fréquemment des imprimés papier [4].

C'est une situation de ce type à laquelle nous nous sommes intéressée dans nos travaux qui sont consacrés à la lecture automatique de tickets de caisse. Ce problème a été proposé par la société AboutGoods (« AboutGoods Company ») dans le but d'obtenir, à très grande échelle, des informations de consommation. La thèse s'est déroulée dans le cadre d'un contrat CIFRE.

La lecture automatique de tickets de caisse pose un certain nombre de difficultés. D'abord, la qualité d'impression est assez variable, avec parfois un faible contraste, des caractères mal imprimés. Ensuite, le stockage du ticket avant sa numérisation (dans un portefeuille, au fond d'une poche...) entraîne des dégradations importantes (salissures, pliures, froissements, déchirures...). Enfin, la numérisation, faite par le consommateur comme nous le verrons plus tard, n'est pas toujours réalisée dans des conditions contrôlées. Par conséquent, l'éclairage pourra être de mauvaise qualité, un fond d'image prédominant avec d'autres éléments pourra perturber la détection du ticket, etc. Ainsi, un problème a priori simple et relevant plutôt d'un travail d'ingénieur se transforme en une réelle problématique de recherche demandant la mise en place de nouveaux outils.

1.2. Les informations de consommation

La connaissance des données de consommations d'une population ou d'un individu est une source d'informations très riche. Elle permet de comprendre le comportement d'un ensemble de consommateurs ou d'un consommateur particulier et présente alors un grand intérêt d'un point de vue socio-économique.

1.2.1. Un intérêt socio-économique

Connaître le comportement de consommateurs présente un intérêt économique évident pour les entreprises du secteur de la vente. Ceci peut se voir à deux niveaux. D'abord à un niveau global, c'est-à-dire au niveau d'une population de consommateurs, avoir des informations statistiques précises sur les achats faits par cette population permet de détecter des tendances, d'anticiper des changements de comportement et ainsi de guider le choix de stratégies marketing, le lancement de nouveaux produits... Au niveau d'un consommateur, si l'on dispose de données sur une période suffisante, cela permet d'avoir une connaissance fine des comportements individuels, et d'avoir alors des actions commerciales beaucoup plus ciblées. Il est ainsi possible de distinguer un achat par habitude d'un achat impulsif et permet alors, par exemple, de proposer des coupons de réduction ciblés de manière personnalisée. On peut noter que de telles stratégies sont déjà utilisées dans les achats en ligne où nos comportements sont enregistrés, analysés et exploités pour nous inciter à de nouveaux achats.

La connaissance du comportement des consommateurs présente également un intérêt sociétal. Là aussi, cet intérêt peut se décliner de manière globale ou individuelle. Dans le domaine de l'alimentation, à un niveau global, la connaissance des achats de denrées alimentaires est une source d'information précieuse pour le baromètre « Sante Nutrition »[1] que le ministère de la Santé a mis en place en 1996. À un niveau individuel, les informations de consommation sont également des indicateurs riches qui reflètent assez finement les habitudes alimentaires d'une famille ou d'un individu et permettent de contribuer au contrôle individuel d'une bonne alimentation. Évoqué pour l'alimentation, ce mécanisme de surveillance peut être étendu à d'autres secteurs comme l'électronique, l'habillement...

1.2.2. Un accès difficile aux informations de consommation

Bien sûr, pour que ces mesures de comportement soient fiables, elles doivent reposer sur des données recueillies en très grand nombre. Ceci est une vraie difficulté car, bien que chaque enseigne enregistre et exploite toutes ces données, dans un monde concurrentiel, elle les conserve jalousement. L'accès par des tiers à ces données doit alors passer par des sollicitations directes des consommateurs en utilisant différents moyens : campagnes de sondage numériques ou téléphoniques, recrutement de panélistes (individus payés pour déclarer leurs achats et équipés de matériels dédiés)... Ces méthodes présentent de fortes

1. http://inpes.santepubliquefrance.fr/Barometres/barometre-sante-nutrition-2008/index.asp

contraintes et/ou un coût important. Par exemple, dans le domaine de l'alimentation, les panélistes sont munis d'un matériel dédié (une douchette et une borne installées à domicile) et doivent suivre une procédure assez lourde consistant à scanner des codes-barres de chaque produit et remplir des formulaires pour justifier/expliquer leurs achats. Toutes ces contraintes techniques, leur coût ainsi que leur représentativité limitée justifient donc la recherche de nouvelles solutions.

1.2.3. Les nouveaux consommateurs

Depuis quelques années, dans le domaine de la consommation, beaucoup d'applications mobiles ont vu le jour. Elles permettent à leurs utilisateurs de scanner les codes des produits de grande consommation, de comparer les prix dans divers magasins, de trouver des informations complémentaires à celles du packaging, de rechercher les magasins qui les vendent, d'inclure ces produits dans des listes de courses ou même de les commander directement sur des sites internet concurrents après les avoir vus en magasin. Avec l'adoption de ces nouveaux outils, de nouveaux comportements se sont très rapidement créés, prenant la forme d'une sorte de communication interactive entre le consommateur et le vendeur. On peut donc envisager de proposer des outils permettant à la fois de répondre au besoin des consommateurs tout en récoltant des informations de consommation.

1.2.4. Le ticket de caisse, une alternative pour une analyse à grande échelle

Pour éviter une trop forte sollicitation du consommateur, une solution relativement simple consiste à exploiter les tickets de caisse. Le problème qui se pose alors est d'amener les consommateurs à transmettre volontairement leurs tickets. Les nouveaux comportements des consommateurs, évoqués ci-dessus, ouvrent la voie à des solutions basées sur des applications utilisables sur téléphone mobile. On peut ainsi imaginer, après un achat, de prendre une photo de son ticket de caisse puis de la transmettre à un serveur. Cette solution, simple, ouvre alors l'accès à une multitude de tickets de caisse et permet d'envisager des analyses sur une bien plus grande échelle que celle permise par le recours à des panélistes. Pour qu'une solution de ce type puisse avoir du succès auprès des consommateurs, il faut bien sûr que ceux-ci y trouvent un intérêt. Dans cette optique, la société AboutGoods a développé une application mobile, nommée TachetKoa?, qui permet de créer simplement une liste de courses, de la partager en temps réel avec les personnes de son choix. Une fonctionnalité complémentaire, en cours de test, a été récemment ajoutée. Elle consiste à proposer à l'utilisateur de prendre une photo de son ticket et de la transmettre à AboutGoods. En retour, sous certaines conditions, il reçoit des informations statistiques sur sa consommation. Ces informations sont pour le moment générées à partir d'une analyse manuelle du ticket, ce qui limite son déploiement, et une des retombées de cette thèse sera de rendre cette analyse automatique.

1.3. Le sujet de la thèse et l'organisation du mémoire

Dans ce contexte, le but de ces travaux de thèse est de définir de nouvelles méthodologies permettant d'extraire et d'analyser l'information contenue dans un ticket de caisse à partir d'une photo prise par un téléphone mobile. Ces développements sont faits dans un contexte industriel imposant un certain nombre de contraintes fortes. En particulier, le système construit doit d'une part être capable de gérer une très grande quantité de tickets et, d'autre part, les résultats fournis doivent être d'une grande fiabilité. Plus précisément, le taux de « faux positifs » doit rester très faible (il est préférable de ne pas reconnaître un produit plutôt que de le confondre avec un autre).
L'originalité des travaux développés tient en plusieurs points :

- dans l'application elle-même, car, à notre connaissance, il n'existe pas de solutions logicielles permettant une lecture automatique complète d'un ticket de caisse ;
- dans la mise en place de l'ensemble de la chaîne de traitement, depuis la saisie de l'image jusqu'à l'identification précise des achats, en travaillant aussi bien au niveau pixel qu'au niveau de la terminologie du domaine ;
- d'un point de vue scientifique, dans l'utilisation, dans la chaîne de traitement, de briques redondantes associant des technologies récentes basées sur le Deep Learning et des technologies plus traditionnelles de manière à ce que la redondance de ces briques renforce la fiabilité.

La suite de ce document s'organise de la manière suivante :

Au chapitre 2, nous présentons le cadre général et la problématique de ces travaux de recherche. Après une rapide étude de l'existant, nous évaluons la faisabilité d'un tel système au travers de la réalisation d'un démonstrateur, afin de mettre en évidence les verrous technologiques. Nous finissons en décrivant la chaîne de traitement envisagée.

Le chapitre 3 est un état de l'art des différentes thématiques liées au problème abordé, à savoir : Les approches deep learning pour la classification d'images et la détection d'objets, la détection de logo, la détection et la reconnaissance de zones de textes et enfin l'analyse sémantique.

Dans le chapitre 4 nous nous intéressons au pré-traitement de l'image avant l'analyse du ticket. L'objectif du pré-traitement est de conserver que l'information essentielle de l'image (le ticket) et d'extraire un maximum de connaissance a priori du contexte. Les différentes étapes du pré-traitement sont présentées. La première étape consiste en la détection du ticket dans l'image. La deuxième étape est la localisation précise du ticket dans l'image. La dernière étape consiste à la détermination de l'enseigne. Finalement, différentes expériences sont faites afin d'évaluer les différentes étapes de cette phase de pré-traitement.

Au chapitre 5 nous proposons une solution pour l'analyse du contenu du ticket de caisse. Cette solution basée sur les méthodes récentes de segmentation sémantique avec

des réseaux de Deep Learning, vise à segmenter les différentes zones du ticket (entête, blocs produits, etc.) en fonction de leur contenu sémantique. Nous investiguons également sur la réalisation d'un OCR spécifique aux tickets de caisse. Finalement, nous proposons des méthodes de construction de base de connaissances pour l'analyse sémantique du contenu des tickets.

2. Objectifs Industriels et Verrous Scientifiques

Ce chapitre présente le cadre général et la problématique de cette thèse. Dans un premier temps, nous commençons par expliquer les objectifs et le contexte industriel du travail envisagé ainsi que l'analyse de l'existant. Puis nous présentons la réalisation d'un démonstrateur dont un des buts est de mettre en évidence les verrous scientifiques du problème posé. Finalement, nous décrirons la chaîne de traitement envisagée pour répondre aux objectifs.

2.1. Objectifs

Comme nous l'avons expliqué dans le chapitre précédent, la finalité de notre travail est de développer une application permettant la lecture automatique de tickets de caisse. Plus précisément, à partir d'une image telle que celle présentée en figure 2.1, nous cherchons à extraire :

— le point de vente ;
— la date et l'heure d'achat ;
— le nombre de produits achetés ;
— l'identification fiable de chaque produit et la quantité achetée ;
— le prix de chaque produit et les remises éventuelles.

Cette problématique fait apparaître deux difficultés. La première est liée à la lecture du texte contenu dans le ticket. La seconde est liée à la compréhension du texte lu, et en particulier à l'identification fiable d'un produit. En effet, les libellés des produits sur les tickets apparaissent généralement sous une forme abrégée, dit libellé court, parfois peu explicite. Et aucune règle standard ne permet de passer d'un libellé court au libellé complet d'un produit.

2.1.1. Analyse et limites des solutions voisines

Actuellement, beaucoup d'applications mobiles proposent aux utilisateurs de prendre en photo leurs tickets de caisse (voir fig. 2.2). À travers l'analyse des tickets, ces applications proposent plusieurs services :

— pour la plupart des applications, l'utilisation des images des tickets sert à offrir des coupons de réductions, des remboursements, des points de fidélité... Pour cela,

FIGURE 2.1. – Exemple d'informations à extraire sur un ticket de caisse.

ces applications, telles que FidMarques [1], Quoty [2], C.wallet [3], Shopmium [4], Plyce [5], etc. se limitent à la reconnaissance d'un seul produit et éventuellement d'une date. La lecture et la compréhension de l'intégralité du ticket ne sont pas nécessaires ;

— d'autres applications, telles que Skerou et Tachetkoa ? [6], proposent de lister tous les produits et leur prix afin de reconstituer numériquement les listes d'achats du consommateur, et/ou de proposer des outils de gestion de dépenses ;

— une autre catégorie d'applications, comme Ticket Tak [7], ProOnGo [8], où JotNot Scanner [9] permet de faire de la gestion de notes de frais. L'image du ticket de caisse sert alors de preuve d'achat, mais c'est à l'utilisateur de renseigner dans un formulaire la date d'achat, l'enseigne et le montant total de ses dépenses.

Malheureusement, pour toutes ces applications, aucune information sur les méthodologies et les techniques mises en œuvre n'est disponible. Aussi, seules les fonctionnalités ont pu être testées. À travers nos tests, il semble acquis qu'aucune de ces applications ne dispose d'une analyse totalement automatique de l'image de ticket de caisse, sauf si le ticket est très simple à lire. Une intervention humaine est souvent nécessaire, ce qui représente un coût non négligeable et un temps d'analyse important.

1. FidMarques
2. Quoty
3. C-Wallet - Apple
4. Shopmium
5. Plyce
6. Tachetkoa?
7. Ticket Tak
8. ProOnGo
9. JotNot Scanner

FIGURE 2.2. – Différentes applications proposant de prendre en photo un ticket de caisse.

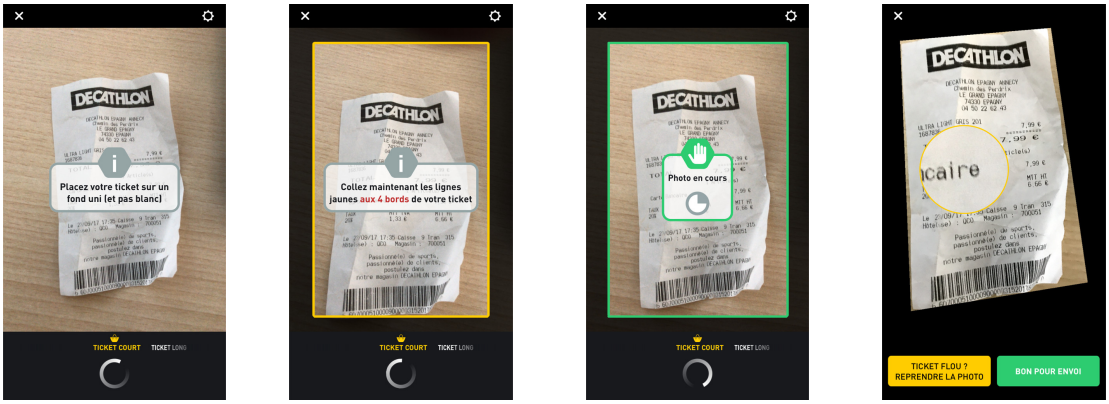
Le fonctionnement de ces différentes applications étant sensiblement identique, nous nous sommes plus particulièrement intéressée à deux d'entre elles : FidMarques et Skerou.

2.1.1.a. Ergonomie des applications

La saisie de l'image du ticket se fait à partir d'un smartphone. Elle est guidée comme cela est illustré sur la figure 2.3 pour l'application FidMarques et se déroule en plusieurs étapes.

Les consignes d'acquisition sont claires : il est demandé à l'utilisateur de placer le ticket sur un fond uni non blanc. Ensuite un algorithme cherche à détecter les bords du ticket et demande à l'utilisateur de faire en sorte d'aligner les quatre bords du ticket aux lignes détectées (cadre jaune sur la figure 2.3). L'algorithme valide alors les conditions d'acquisition, un message s'affiche pour informer l'utilisateur et le déclenchement de la prise de photo se fait automatiquement. La dernière étape consiste à demander à l'utilisateur de vérifier la lisibilité du ticket. Quand l'utilisateur valide cette lisibilité, l'image acquise est transmise à un serveur pour qu'elle soit analysée. Le résultat est restitué au bout de quelques minutes voir quelques heures.

L'acquisition guidée est plutôt bien maîtrisée, mais elle demande une forte participation de l'utilisateur. Elle peut également prendre du temps puisque l'utilisateur n'a pas la main sur le déclenchement de l'acquisition. Si un défaut dans les conditions de prise de vue persiste, il est impossible pour l'utilisateur de pouvoir envoyer une photo puisque



(a) Les consignes d'acquisition sont données à l'utilisateur.

(b) L'algorithme donne les indications nécessaires pour valider l'acquisition.

(c) Réalisation de l'acquisition si les conditions sont jugées bonnes (cadre vert).

(d) L'algorithme demande à l'utilisateur de valider la lisibilité de l'image acquise.

FIGURE 2.3. – FidMarques : acquisition guidée.

l'algorithme refusera de déclencher l'acquisition. De plus, le contexte imposé est souvent trop strict pour pouvoir correspondre à une prise de photo en magasin, par exemple au moment où l'utilisateur récupère son ticket, instant où le ticket a plus de chance de ne pas avoir subi de dégradations.

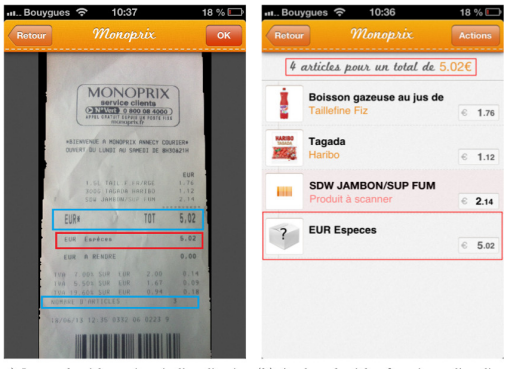
2.1.1.b. Analyse de la liste d'achats

Concernant l'analyse du contenu du ticket, les tests effectués révèlent que le traitement OCR réalisé par ces applications est globalement satisfaisant. Sur une quinzaine de tickets de caisse, avec Skerou, où la reconnaissance de caractères est très probablement automatique, le taux de bonne détection (pourcentage de caractères bien lus que nous avons mesuré est supérieur à 91%).
Le nombre d'erreurs est par contre plus grand lorsque les applications souhaitent proposer une liste des produits achetés en interprétant les libellés courts (nomination abrégée des produits). Toujours sur Skerou, la reconnaissance des produits est essentiellement gérée par apprentissage. C'est-à-dire que si l'application rencontre un libellé court inconnu, le consommateur est invité à scanner le code barre correspondant, ce qui permet à l'application de lier le libellé court au produit une fois pour toutes. Le code barre du produit correspond à son GTIN (Global Trade Identification Number). Composé de 13 chiffres, il identifie un produit de manière unique. Cette phase d'apprentissage est bien sûr assez longue. De plus, elle ne présente pas de caractère générique, car les libellés courts d'un même produit varient selon les enseignes, et même parfois à l'intérieur d'une

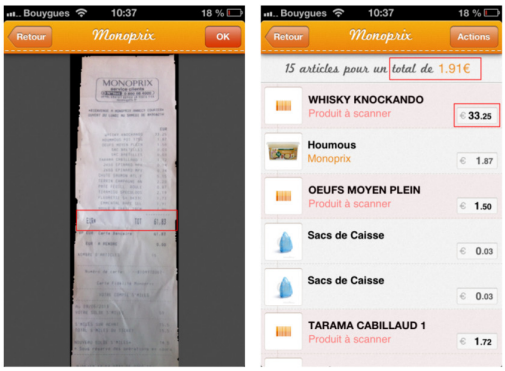
même enseigne. Par exemple le produit « poulet » pourra apparaître dans les libellés courts sous les formes : « poulet », « plt », « plet ».
.

2.1.1.c. Analyse de la cohérence du ticket

Nous avons également rencontré parfois des incohérences dans les informations extraites des tickets de caisse.



(a) Incohérence sur le nombre d'articles présents dans le ticket de caisse.



(b) Incohérence sur le montant total du ticket de caisse.

FIGURE 2.4. – Skerou : incohérences dans l'analyse du contenu des tickets de caisse.

Par exemple, sur la figure 2.4a (résultat obtenu avec Skerou), l'application a assimilé la ligne qui indique que le client a payé en espèces avec un produit coûtant 5.02 euros, montant correspondant au total des achats. Cette erreur pourrait être corrigée en ajoutant une analyse sémantique des résultats obtenus. En effet, quelques lignes plus bas (encadré bleu sur l'image de gauche), il est écrit clairement que le nombre d'articles est de trois et non de quatre et le total est vérifiable en sommant simplement les prix des trois articles. Une autre erreur, du même type, est illustrée sur la figure 2.4b, où le montant total (1.91 euros) est incohérent étant donné les prix des articles dans le ticket (le premier article a une valeur de 33.25 euros).

2.1.1.d. Synthèse sur l'analyse des applications existantes

Aucune des applications testées ne correspond au besoin de la société AboutGoods : lecture partielle, intervention humaine, taux d'erreur trop important... L'application Skerou n'est d'ailleurs plus disponible aujourd'hui. L'application que nous envisageons doit répondre à de fortes contraintes : niveau de confiance élevé sur chacune des informations extraites, capacité à analyser tout type de ticket de caisse, à pouvoir identifier tous les produits achetés et à traiter des acquisitions difficiles (ticket froissé...).

2.1.2. Les travaux scientifiques sur l'analyse de tickets de caisse

Dans la littérature, il y a peu de travaux spécifiquement dédiés à l'analyse des images de tickets de caisse.

Dans [10], les auteurs abordent la lecture automatique de petits documents (tickets de caisse, tickets de train, reçus de parking...). La méthodologie proposée est essentiellement illustrée par l'analyse de tickets de caisse. Elle comporte une première étape de normalisation : seuillage grossier et clustering sur les composantes connexes issues du seuillage (les auteurs affirment que généralement le cluster le plus important correspond à du bruit et le suivant aux caractères), redressement vertical du ticket, élimination du fond, ajustement de la luminance du ticket et enfin binarisation fine. La seconde étape est l'extraction des informations utiles. Elle débute par une reconnaissance des caractères (avec un OCR du commerce non précisé) et une analyse des résultats obtenus en recherchant des expressions régulières (motifs caractéristiques d'informations telles qu'une date, un numéro de téléphone...) et des modèles de voisinage entre informations extraites (par exemple un montant doit se trouver à droite de *Total*). La dernière étape utilise un système expert dont le rôle est de distinguer les différents types de petits documents que ce système est capable de gérer. Aucune étude de performance n'est proposée.

Dans [11], les auteurs partent du constat que la spécificité des tickets de caisse (fontes particulières, proximité des caractères) demande une adaptation des outils existants. Ils supposent que la numérisation du ticket est suffisamment propre pour permettre une segmentation facile et performante des lignes de texte. Ils se concentrent alors sur la segmentation et la reconnaissance des caractères sur une même ligne. La segmentation est effectuée en utilisant la méthode des forêts aléatoires. La reconnaissance est réalisée par un classifieur SVM avec un noyau RBF. Les performances obtenues sur une vingtaine de tickets sont bonnes (voisines de 91%). Malheureusement la base utilisée et les données d'expérimentations ne sont pas disponibles pour pouvoir se comparer.

Dans [12] les auteurs s'intéressent à une situation ou plusieurs tickets sont scannés simultanément sur un fond blanc. La difficulté est alors de séparer chacun des tickets. Les auteurs appliquent d'abord un détecteur de contours et définissent un ensemble de boîtes englobant des caractères ou des groupes de caractères. Ensuite, un clustering spectral appliqué sur les distances entre ces boîtes englobantes permet de séparer chacun des tickets. La lecture de chaque ticket est alors effectuée en utilisant un OCR du commerce (Tesseract). Avec des conditions d'acquisition très favorables, les performances de clustering sont bonnes (93% de bonne détection) mais les performances de lecture des caractères ne sont pas fournies.

Une étude détaillée du processus complet d'analyse d'un ticket de caisse est proposée dans [13]. La méthode se décompose en quatre étapes principales :

— la localisation du ticket dans l'image. Les auteurs comparent plusieurs approches : des méthodes simples basées sur la binarisation, l'utilisation d'un réseau neuronal convolutionnel (entraîné à détecter les angles du ticket) et d'un classifieur Haar

Cascade [14]. Le réseau neuronal atteint de bonnes performances, mais est sensible à la position du texte par rapport au bord du ticket. De la même manière, la binarisation adaptative donne de très bons résultats à condition que le fond de l'image soit très peu texturé et homogène. Quant aux classifieurs Haar Cascade, malgré un long temps d'entraînement, les résultats obtenus sont peu satisfaisants ;
— la localisation des caractères dans l'image binaire du ticket. Les auteurs choisissent d'utiliser la recherche de composantes connexes appliquée sur les contours de l'image du ticket. Un filtrage est également nécessaire pour éliminer les composantes correspondant à du bruit. Les caractères peuvent ensuite être regroupés en mots par association de caractères voisins ;
— la reconnaissance des caractères. Plusieurs méthodes ont été testées : la reconnaissance de caractères en entraînant des réseaux de neurones, mais cette méthode reste très sensible à la qualité de la segmentation isolant les caractères. La solution alternative choisie est la reconnaissance des mots complets avec l'utilisation d'un réseau LSTM (Long Short-Term Memory), autrement dit un réseau de neurones récurrents. Les performances obtenues sont peu satisfaisantes ;
— l'extraction des informations contenues dans le ticket. Cette dernière étape utilise simplement des expressions régulières. Par exemple, les prix de chaque produit sont obtenus en recherchant une chaîne de caractère correspondant au format XX.XX ou X est un chiffre.

Les auteurs concluent leur article en notant que le problème de la lecture d'un ticket de caisse est un problème complexe fortement dépendant de la qualité des images analysées. En particulier les caractères jointifs ou cassés sont une des principales sources d'erreur.

Dans [15] la lecture automatique d'un ticket de caisse est décomposée en différentes étapes : une binarisation suivie d'une dilatation morphologique, une détection de composantes connexes (supposées correspondre aux zones de texte), un filtrage des composantes ne correspondant pas à du texte par analyse statistique de paramètres de texture, une normalisation de manière à disposer de caractères ou de blocs de caractères horizontaux et enfin une lecture du texte effectuée par un OCR du commerce (Tesseract). Ces travaux supposent disposer d'une image ne contenant qu'un ticket et la base sur laquelle les tests sont effectués n'est pas détaillée ni disponible et ne comporte que 68 tickets. Ces travaux, publiés en 2017, sont trop récents pour avoir pu guider la construction de notre démonstrateur. Néanmoins, nous les mentionnons, car ils confortent nos conclusions.

De l'analyse de ces quelques articles dédiés à la lecture de tickets de caisse, nous pouvons tirer les conclusions suivantes :
— d'abord, si les stratégies sont voisines, en particulier dans l'enchaînement des étapes principales, les méthodologies utilisées à chaque étape diffèrent sensiblement d'un article à l'autre ;
— ensuite, dans la mesure où les solutions proposées n'utilisent pas de base commune, il est impossible de les comparer, et en particulier de savoir quelles méthodologies sont les plus performantes ;
— enfin, les méthodologies supposent souvent disposer d'images de bonne qualité et

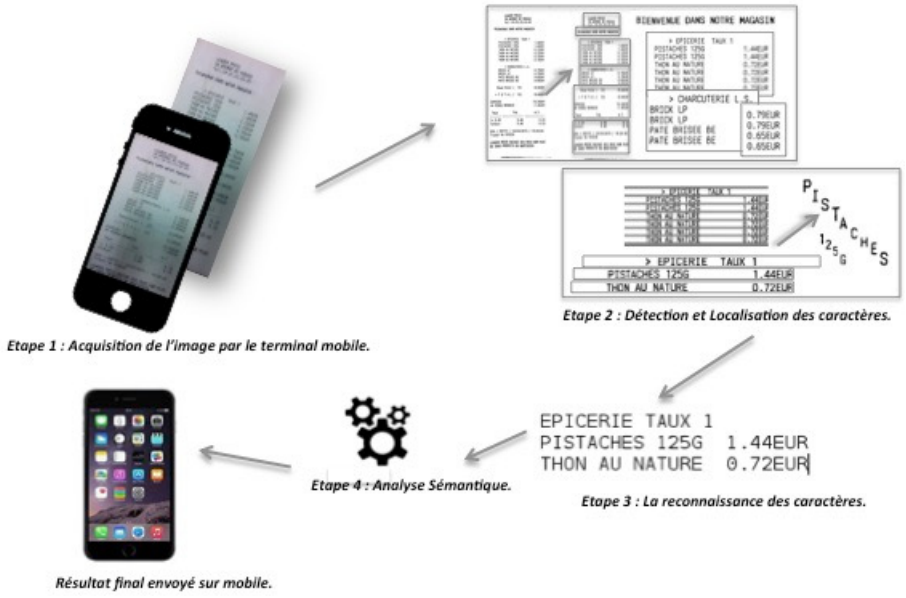


FIGURE 2.5. – Les quatre étapes principales du démonstrateur de lecteur de ticket de caisse.

les performances se dégradent rapidement si cette qualité n'est pas suffisamment assurée.

2.2. Construction d'un démonstrateur

2.2.1. Objectif

AboutGoods étant une jeune entreprise, il était indispensable de pouvoir proposer très rapidement un premier système de lecture automatique de tickets de caisse, jouant le rôle de démonstrateur, afin d'appuyer la stratégie de l'entreprise auprès d'investisseurs et de clients et de pouvoir ainsi pérenniser l'activité. D'un point de vue scientifique, ce démonstrateur a également permis de mettre en évidence les verrous scientifiques d'un tel projet et de guider le choix des méthodologies.

2.2.2. Structure et contraintes du démonstrateur

La nécessité du développement rapide d'un démonstrateur, à coût réduit, nous a amenée à nous placer dans des conditions plutôt favorables, avec des tickets de caisse bien conservés, très peu détériorés, et des acquisitions de bonne qualité. La structure de notre démonstrateur fait alors apparaître quatre étapes principales, comme on peut le voir sur la figure 2.5.

1. L'acquisition de l'image par un terminal mobile. Cette étape est très importante, elle doit assurer une qualité d'image suffisante pour permettre l'analyse du contenu

de l'image. Elle comprend :
— une aide à l'utilisateur afin d'acquérir une image de bonne qualité. Cela se traduit par la pré-détection des bords du ticket et la superposition à l'image d'une boîte englobante incitant l'utilisateur à faire coïncider les bords du ticket avec la boîte englobante ;
— l'acquisition proprement dite d'une image, déclenchée par l'utilisateur.

Les étapes suivantes sont réalisées sur un serveur distant.

2. La localisation du texte. Cette étape tirera parti des conditions favorables de l'acquisition pour localiser des blocs, des lignes ou des groupes de caractères qui seront localisés par une boîte englobante ;
3. La reconnaissance des caractères : plusieurs technologies ont déjà fait leurs preuves. Il s'agit de choisir la technologie la plus adaptée et la plus favorable à notre situation, notamment en termes de compromis qualité de résultat / temps de traitement / coût ;
4. L'analyse sémantique qui permet de corriger certaines erreurs et d'interpréter les résultats fournis par l'OCR.

La nécessité de l'obtention rapide d'un premier démonstrateur n'exclut pas d'imposer un certain nombre de contraintes. D'abord, l'étape de prise de photo doit être fluide, rapide, simple pour l'utilisateur, tout en assurant une qualité d'image suffisante pour permettre l'exploitation par l'algorithme de lecture du ticket. Ensuite, parmi les informations qui doivent être extraites du ticket (voir figure 2.1), la non détection doit être privilégiée par rapport à l'erreur de détection. Ainsi, si nous prenons comme exemple l'information « Point de vente » il faut que l'algorithme soit certain de l'enseigne lue sinon il est préférable de ne pas donner d'information. D'un point de vue commercial, il est en effet préférable qu'un ticket soit déclaré non lisible plutôt que d'annoncer une information erronée. Une reprise manuelle peut d'ailleurs être envisagée sur les tickets non lus, ou éventuellement demander à l'utilisateur de faire une nouvelle acquisition dans de meilleures conditions.

2.2.3. Quelques considérations de résolution spatiale

Les données à analyser proviennent essentiellement de terminaux mobiles et les résolutions des images sont très variées en fonction des modèles de smartphone utilisé. Les images de meilleure qualité atteignent une résolution de 12Mpx, mais la résolution d'images la plus fréquente est de 8Mpx ce qui reste une qualité tout à fait acceptable puisque dans ces images les caractères ont une taille moyenne de 24x32 pixels. Jusqu'à 5Mpx, les images reçues restent exploitables puisque la taille moyenne des caractères est de 14x24 pixels. En dessous, la lisibilité du ticket dans l'image est incertaine.

Une autre information à prendre en considération est la distance de prise de vue de l'image qui est généralement directement associée à la taille du ticket. En effet, plus le ticket est long, plus la distance de prise de vue sera grande afin d'assurer la présence du ticket en entier dans l'image. La résolution des caractères est alors fortement diminuée

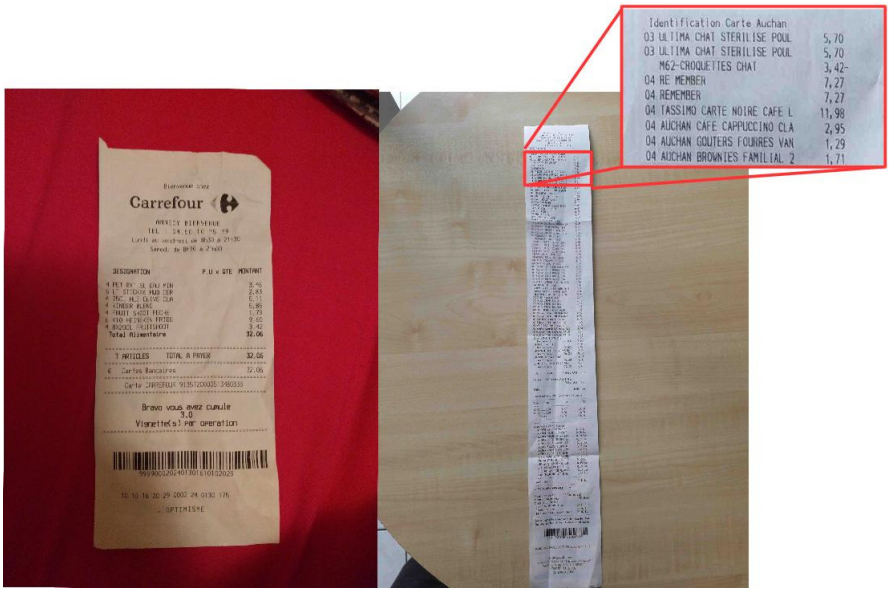


FIGURE 2.6. – Différence entre un ticket à la limite de la taille permettant la lisibilité et un ticket de taille moyenne.

puisqu'elle peut typiquement être divisée par deux. Un ticket long est un ticket correspondant à environ 120 lignes de texte, comme on peut le voir sur la Fig.2.6. Ce ticket occupe 20% de la largeur totale et 90% de la longueur totale de l'image de 8Mpx. Il correspond à la limite que nous avons fixée afin d'assurer la lisibilité d'un ticket sur une simple photo. Dans le cas de tickets très longs pour lesquels il est impossible d'assurer une taille de caractère suffisante, nous envisageons de proposer une prise de vue en vision panoramique.

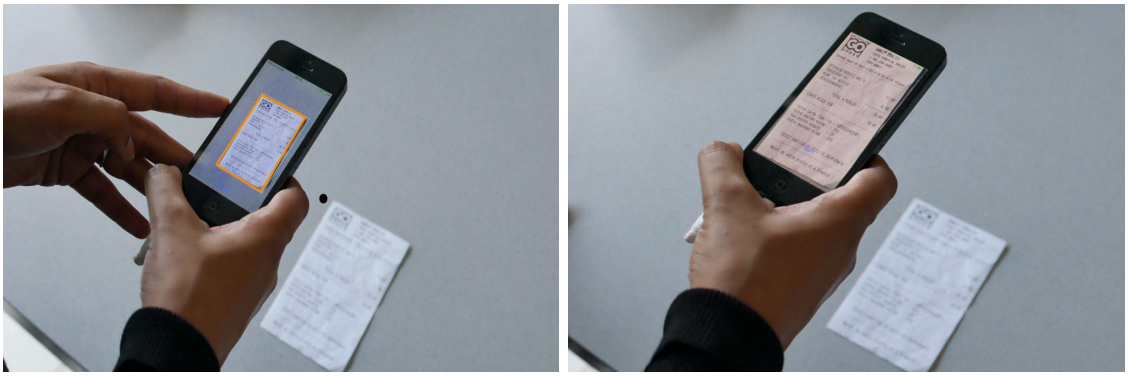
2.2.4. Étape 1 : acquisition Guidée

Dans le domaine de l'analyse d'image, il est bien connu que l'acquisition est une étape cruciale qui impacte fortement la suite des traitements. C'est pourquoi nous avons fait le choix d'accompagner l'utilisateur lors de la saisie de l'image en localisant le ticket en temps réel. Rappelons que cette étape a pour objectif d'assurer une qualité d'image suffisante pour permettre l'analyse, mais sans trop contraindre l'utilisateur.

2.2.4.a. Description globale du scénario d'acquisition

Dès que l'utilisateur positionne son smartphone au-dessus du ticket, un algorithme cherche à détecter les bords du ticket et superpose à l'image un cadre orange, incitant l'utilisateur à faire coïncider les bords du ticket avec ce cadre orange. Un exemple de résultat obtenu est présenté en Fig.2.7a. Une fois que l'utilisateur juge que le ticket a bien été localisé, il choisit de déclencher l'acquisition (bouton « capture »). L'image acquise est alors ré-affichée sur l'écran du téléphone (Fig.2.7b) permettant ainsi d'assurer

un contrôle de l'utilisateur sur la qualité de l'image acquise avant que celle-ci ne soit envoyée à un serveur pour être analysée.



(a) Détection en temps réel du ticket visible par un rectangle orange.

(b) Affichage de l'image du ticket transmise au serveur.

FIGURE 2.7. – Différentes étapes pour l'acquisition guidée de l'image par le terminal mobile.

2.2.4.b. Détecteur spécifique de ticket de caisse

L'objectif est de détecter les bords du ticket en temps réel afin de guider l'utilisateur et d'obtenir une image rognée limitée au seul ticket. Les détecteurs de contours classiques ne sont pas adaptés à notre situation, car du fait du contenu même du ticket, ils génèrent de nombreux contours parasites. Nous avons donc mis au point un détecteur spécifique adapté à la détection des quatre bords du ticket. Ce détecteur part de l'hypothèse que le ticket est bien orienté (bords pratiquement parallèles aux bords de l'image) et que les bords du ticket sont clairs sur un fond uniforme plus sombre que le ticket. Ce détecteur spécifique de ticket de caisse constitue une première contribution dans ces travaux de thèse.

Nous avons défini quatre masques (un pour chaque bord) illustrés sur la figure 2.8, non symétriques, sur lesquels sont définies trois zones. Une zone claire qui correspondrait au ticket, une zone sombre qui correspondrait au fond et une zone centrale, neutre, d'une certaine largeur, permettant une tolérance à la non-verticalité d'un ticket.

Le fonctionnement du détecteur est expliqué pour le bord gauche, l'extension aux autres bords étant immédiate. Si l'on note μ_i et σ_i la moyenne et la variance de la zone i, notée z_i par la suite, un pixel appartiendra au bord gauche si :

— $\mu_1 < \mu_{1+3} + \Delta_1$: la zone 1 est bien plus sombre que la zone $(1 + 3)$;
— $\mu_3 > \mu_{1+3} + \Delta_2$: la zone 3 est bien plus claire que la zone $(1 + 3)$;
— $\sigma_3 < Th$: la zone 3, qui devrait correspondre au bord du ticket, est homogène.

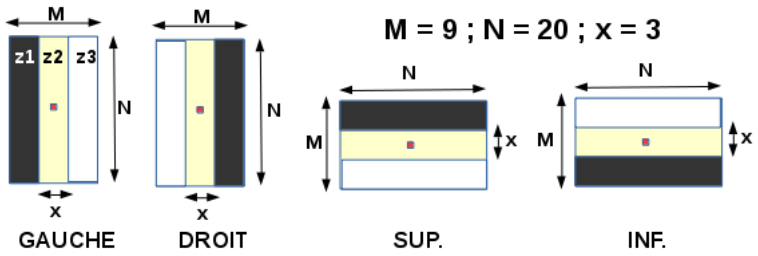


FIGURE 2.8. – Masques pour la détection des bords du ticket.

FIGURE 2.9. – Résultats de la détection des points contours sur une image.

où Δ_1, Δ_2 et Th sont des constantes. Pour éviter les fausses et les doubles détections, nous ne gardons qu'un point par ligne, celui répondant le mieux aux critères choisis, la meilleure réponse étant définie comme la plus grande différence des moyennes des zones 1 et 3 (max de $\mu_3 - \mu_1$). Pour gagner en rapidité, nous ne traitons qu'une ligne sur trois (choix empirique permettant un gain de temps sans dégradation visible des performances). Nous obtenons ainsi un ensemble de « points contours » pour chaque bord du ticket. La figure 2.9 donne un exemple de résultat. Une fois les quatre ensembles de points de contours obtenus, quelques considérations géométriques simples permettent d'obtenir les droites correspondant aux quatre côtés du ticket puis le rectangle englobant au mieux le ticket.

Cet algorithme fait intervenir un jeu de 7 paramètres : la composante couleur utilisée (R,V ou B), la hauteur M et la largeur N des masques, la largeur x de la zone neutre, les écarts d'intensité Δ_1, Δ_2 et le seuil Th sur l'écart-type.

2.2.5. Étape 2 : Prétraitement et localisation des caractères

Grâce à l'acquisition guidée, l'image transmise au serveur ne contient que le ticket à analyser. Dans ce paragraphe, nous allons présenter les méthodes simples que nous avons mises en place pour localiser et reconnaître les zones de texte. Cette démarche comporte

trois étapes : une binarisation, un filtrage éliminant le bruit de binarisation et enfin la localisation des caractères.

2.2.5.a. Binarisation et « nettoyage » de l'image

L'image que nous devons analyser comporte des caractères sombres sur un fond clair. C'est donc une situation tout à fait favorable à la binarisation, pour peu que les conditions d'éclairage de l'acquisition soient bonnes. Dans l'élaboration de ce démonstrateur, nous nous plaçons dans des conditions favorables et nous supposerons donc que l'hypothèse de bonnes conditions d'éclairage est satisfaite.

Il existe de très nombreuses méthodes de binarisation. On peut d'abord distinguer :

— la binarisation globale : elle considère l'image tout entière pour définir un seul et unique seuil appliqué sur toute l'image ;

— la binarisation locale adaptative : pour chaque pixel, le seuil est choisi en fonction de la distribution des intensités au voisinage du pixel considéré.

L'approche adaptative, bien que plus lourde en termes de calcul, est plus appropriée dans notre situation. En effet, même si les conditions d'éclairage sont supposées favorables, dans le cas de tickets de grandes tailles, la luminosité moyenne peut varier de manière importante entre le haut et le bas du ticket et nécessiter une adaptation locale du seuil de binarisation.

Nous avons testé deux méthodes permettant de calculer un seuil local de binarisation. La première consiste à utiliser la binarisation d'Otsu [16] sur des fenêtres glissantes de grande taille. La méthode d'Otsu, bien adaptée à une situation bi-modale comme la nôtre, cherche le seuil minimisant la variance intra-classe de chacune des deux classes et maximisant la variance inter-classe. Cette méthode a fourni de bons résultats, pour peu que la fenêtre de calcul soit suffisamment grande pour assurer de contenir des pixels des deux classes (fond / zone de texte).
La seconde approche de binarisation, disponible dans OpenCV, utilise une moyenne pondérée gaussienne des valeurs des pixels qui se trouvent dans la fenêtre d'analyse. Le seuil est alors déterminé par cette moyenne pondérée à laquelle est soustraite une constante à définir. Avec cette approche, le choix de la taille de la fenêtre de calcul est moins critique, car une zone claire ne contenant que des pixels du fond aboutit malgré tout à un choix cohérent du seuil de binarisation grâce à la soustraction de la constante. De plus, la fenêtre d'analyse contenant généralement plus de pixels du fond que de pixels de caractères, sous réserve d'un choix cohérent de la constante soustraite, le risque de choisir un seuil trop bas érodant les caractères est très limité.

Malgré les bons résultats de la binarisation, il peut persister un bruit de binarisation qui se traduit par la présence de pixels noirs isolés. Plusieurs solutions sont envisageables pour éliminer ce bruit : ouverture morphologique, filtre médian... Nous avons opté pour le filtre médian car, en observant qualitativement les effets de ces deux filtres sur quelques images, il nous a semblé, de manière subjective, que l'effet de filtrage était sensiblement identique, mais que l'ouverture déformait légèrement plus les caractères que le médian.

2.2.5.b. Localisation des caractères

La localisation du texte peut se situer à différents niveaux de granularité : au niveau caractère, groupe de caractères (portion d'une même ligne), ligne ou enfin groupe de lignes, les outils de reconnaissance de caractères acceptant généralement ces différentes formes d'entrée. Dans cette version de démonstration du lecteur automatique de tickets, nous avons cherché à localiser des lignes ou des portions de lignes. La tâche est ici facilitée par nos conditions de prise de vue qui incitent fortement l'utilisateur à réaliser l'acquisition lorsque l'image du ticket est verticale. Ainsi, les lignes de texte peuvent être considérées comme horizontales.

La segmentation de zones de texte est un problème étudié depuis très longtemps dans la littérature. Les travaux présentés ci-dessous décrivent les méthodes pour réaliser la segmentation en caractères, mais les mêmes méthodes sont applicables pour la segmentation en lignes.

Dans [17], état de l'art datant de 1996, on trouve ainsi plusieurs solutions pour la segmentation en caractères de lignes horizontales composées de caractères imprimés. Une première solution procède en deux étapes :

— un premier parcours, de gauche à droite, détecte les espaces blancs séparant chaque caractère, permettant ainsi d'estimer la largeur moyenne d'un caractère. Cette stratégie repose sur l'hypothèse, tout à fait réaliste, que, dans un document imprimé, l'écart entre les caractères est sensiblement constant ;
— un deuxième parcours, de droite à gauche, exploite la connaissance de cette distance moyenne pour définir les points de segmentation.

Toujours dans [17], une seconde méthode utilise des projections verticales. À l'intérieur d'un bloc contenant une ligne, cette méthode consiste à calculer le nombre de pixels noirs de chaque colonne. Sur cette projection, les caractères apparaissent comme des pics et les espaces entre caractères comme de vallées. Une détection pic/vallée permet alors de segmenter les caractères. Les auteurs montrent que cette méthode est capable de gérer de légers chevauchements entre caractères.

Dans [18] les auteurs proposent plusieurs solutions de segmentation des caractères imprimés pour les cas difficiles tels que des caractères en partie effacés ou des caractères collés, situations que l'on peut effectivement rencontrer sur les tickets. Pour les caractères en partie effacés (mais cela englobe la situation des caractères de petite taille tels que virgule ou point), le principe consiste à effectuer des projections multi-lignes. En supposant que les caractères des différentes lignes sont alignés verticalement (ceci est généralement vrai sur les tickets), la faible contribution des caractères effacés est alors compensée par la contribution des autres caractères situés sur la même verticale. Pour les caractères qui se chevauchent, le principe consiste à utiliser des statistiques sur les dimensions des caractères segmentés.

Ces méthodes supposent la plupart du temps la disposition d'une segmentation en lignes. Cette segmentation ne pose pas de difficultés dans les conditions favorables où nous nous plaçons. En effet, l'acquisition contrôlée assure la quasi-verticalité de l'image,

et une simple projection horizontale (comptabilisation du nombre de pixels noirs par ligne) permet de faire apparaître des courbes sur lesquelles les pics correspondent aux lignes et les vallées aux interlignes. Les limites de chaque ligne sont alors obtenues en recherchant simplement le pixel noir le plus à gauche et le pixel noir le plus à droite. Néanmoins, dans la pratique, les déformations du ticket peuvent perturber l'horizontalité des lignes (voir un cas extrême dans le tableau Tab : 2.2) et ne pas permettre une bonne segmentation en lignes. Une manière de contourner cette difficulté consiste à procéder d'abord à des projections verticales sur des groupes de lignes afin de faire apparaître, par le même mécanisme, des blocs de texte (par exemple le bloc des libellés courts, puis à sa droite le bloc des prix). La recherche des lignes par projection horizontale est alors effectuée à l'intérieur de chaque bloc, diminuant ainsi le risque de chevauchement horizontal de deux lignes successives. Une même ligne peut alors être divisée en plusieurs parties. Par exemple, la ligne d'un achat comprendra une première partie de ligne contenant le libellé court et une seconde partie contenant le prix.

2.2.6. Étape 3 : Reconnaissance Optique de Caractères

La Reconnaissance Optique des Caractères (OCR) est un domaine très actif depuis plusieurs décennies. Dans le cas de la reconnaissance de caractères imprimés, plusieurs solutions existent aujourd'hui qui offrent des performances largement satisfaisantes telles que Tesseract [19], ABBYY FineReader [20]. D'autres approches plus récentes, basées sur des réseaux de neurones profonds, telles que Google Vision [21], ont permis d'améliorer encore les performances. Nous avons évalué ces outils de reconnaissances optique de caractères en comparant leur performance sur une vingtaine de tickets, et nous avons également mesuré leurs limites. Les résultats détaillés seront présentés dans la section 2.3.4. Nous nous intéresserons dans cette section aux problèmes soulevés par l'utilisation des OCR dans notre contexte spécifique.

De manière générale les OCRs assignent un niveau de confiance à chaque caractère reconnu et utilisent un dictionnaire pour corriger des erreurs de reconnaissance quand ils le jugent nécessaire. Le dictionnaire correspond à la langue du document, renseignée manuellement par l'utilisateur ou déduite automatiquement par l'OCR.

Il faut noter que, d'un point de vue technique et afin de permettre un bon fonctionnement de ces OCRs, il a été nécessaire d'étendre la boîte englobante de chaque caractère ou de chaque ligne. Par exemple, Tesseract fonctionnant avec des détecteurs de points d'intérêt situés en particulier sur les contours du caractère, une boîte englobante collée au contour extérieur du caractère gêne la détection de ces points d'intérêt. Un bord blanc dont la dimension est égale à 30% de la dimension de la boîte englobante a donc été ajouté sur chaque côté. Un exemple est présenté en Fig.2.10 où le caractère « B » est interprété par Tesseract comme un « @ » sans l'ajout d'un bord blanc, alors qu'il est correctement interprété avec l'ajout du bord blanc. Le choix du coefficient de 30% résulte de tests expérimentaux.

Nos tests ont également révélé que la reconnaissance globale sur une ligne de texte

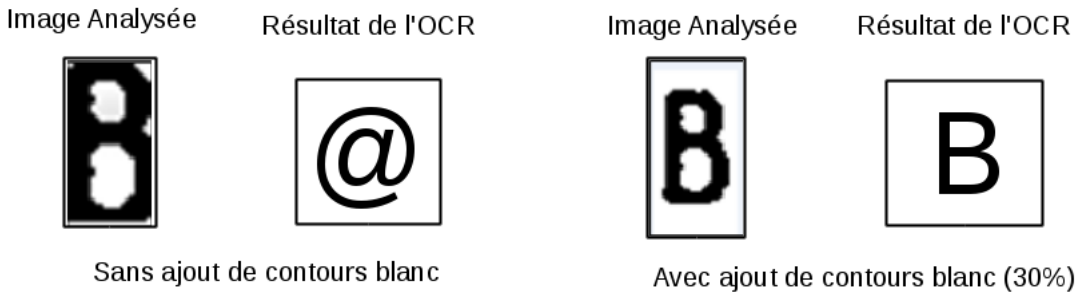


FIGURE 2.10. – Erreur de reconnaissance si la boîte englobante est limitée aux frontières externes du caractère par Tesseract.

FIGURE 2.11. – Correction non faite par Tesseract car le niveau de confiance accordé au caractère erroné est trop élevé.

était généralement plus performante qu'une reconnaissance caractère par caractère. Ceci est dû à l'utilisation par ces OCRs d'un dictionnaire corrigeant d'éventuelles erreurs de reconnaissance faites au niveau caractère. Néanmoins, en analysant de manière plus fine l'utilisation que Tesseract faisait de ce dictionnaire, nous avons pu constater :

— des corrections non effectuées. Ceci se produit lorsque les degrés de confiance sur la reconnaissance des caractères d'un mot sont trop élevés. Dans cette situation, le dictionnaire n'est pas déclenché. Par exemple sur la figure 2.11 le mot « Total » a été interprété par Tesseract en « Tutal » : le caractère « u » a été reconnu avec un taux de confiance trop important pour pouvoir autoriser une correction par le dictionnaire de Tesseract ;
— des corrections erronées. Ceci est dû à l'absence des libellés courts dans le dictionnaire. Par exemple « ALL » (pour allumette) peut devenir « AIL », ou encore « PDT » (pour président) qui peut être corrigé en « POT ».

2.2.7. Étape 4 : Analyse Sémantique

À la sortie de l'OCR, comme nous lui avons fourni en entrée des lignes ou des morceaux de lignes, nous supposons disposer de mots. La dernière étape est alors l'analyse sémantique du texte extrait. Rappelons que cette étape a un double objectif : d'une part la correction de certaines erreurs et d'autre part l'interprétation du résultat de la

reconnaissance de caractères.

Pour la correction, nous avons utilisé un dictionnaire, mais en ayant préalablement construit notre propre dictionnaire adapté à notre problématique, c'est-à-dire limité au lexique des tickets de caisse. La comparaison au dictionnaire, systématique dans notre cas, a été faite en utilisant le modèle des N-grammes [22]. Un N-gramme est une séquence de N caractères consécutifs (typiquement N vaut 2 ou 3). La représentation d'une chaîne de caractères (texte ou mot) se fait alors par l'histogramme de tous les N-grammes de la chaîne. À partir de cette représentation, il est alors possible de calculer un indice de similarité entre deux chaînes de caractères. Dans notre contexte, quand cette similarité entre une chaîne extraite du ticket et un élément du dictionnaire est forte (voisine, mais différente, de 1), on considère qu'il s'agit d'une erreur et l'on effectue une correction. Ici, l'utilisation des N-grammes a été préférée à la distance de Levenshtein pour des raisons de simplicité, de rapidité et de souplesse. En particulier le choix de N permet de contrôler l'importance que l'on veut donner au contexte de notre analyse.

Dans un deuxième temps, à l'aide d'expressions régulières, d'une base de données de connaissances et de lexiques, nous extrayons les informations « de bases » telles que la date d'achat, l'enseigne, la ville ou code postal, le montant total, le nombre d'articles, etc.
Ici on distingue deux types d'informations, les informations globales (enseigne, ville, code postal, numéro de téléphone) qui peuvent se retrouver à l'identique d'un ticket à l'autre et les informations spécifiques (les chaînes telles que date d'achat, montant total, et nombre d'articles). Pour les informations globales, nous avons utilisé des bases de données existantes contenant l'ensemble des villes françaises et leur code postal, ainsi que la liste des enseignes et magasins avec leur numéro de téléphone. Ainsi à l'aide d'expressions régulières et d'un calcul de similarité via la méthode des N-grammes nous sommes capables de retrouver ces informations dans le ticket de caisse. Quant aux informations spécifiques, nous avons dû définir un lexique pour chacune de ces informations. Par exemple pour la recherche du montant total, le lexique contient les expressions suivantes : « TOTAL », « TOTAL A PAYER », « TOTAL A REGLER », « TOT » , « A PAYER », « MONTANT DU ». C'est également la méthode des N-grammes qui a été utilisée pour retrouver ces informations spécifiques.

2.3. Évaluation et mise en évidence des limites du démonstrateur

L'élaboration d'un démonstrateur nous a permis de mettre en évidence les performances et les limites des méthodes utilisées pour chacune des quatre étapes présentées en section 2.2.2.

Avant d'examiner les résultats obtenus pour ces différentes étapes, précisons les bases

sur lesquelles ces expérimentations ont été faites.

2.3.1. Bases d'évaluation

Nous avons évalué les performances de ce premier lecteur automatique de tickets de caisse sur deux bases d'images dont les conditions d'acquisition sont différentes.

Une première base, notée B1 dans la suite, est constituée de 150 images scannées, avec une qualité d'image plutôt correcte (tickets propres et ayant subi peu de dégradations comme on peut le voir sur la Fig. 2.12a). Cependant, ces images, fournies par une société extérieure cliente d'AboutGoods, peuvent contenir des informations supplémentaires qui bruitent et complexifient leur analyse (tampons, libellés produits entourés manuellement, où encore identifiant du ticket en haut de celui-ci).

(a) Exemples de tickets figurant sur la base de test d'images scannées.

(b) Exemples de tickets figurant sur la base de test d'images acquises via différents terminaux mobiles.

FIGURE 2.12. – Les différentes bases de tests sur lesquelles nous avons évalué les performances des différentes étapes du lecteur automatique du ticket de caisse.

La deuxième base, notée B2, a été réalisée par nos soins. Elle est composée d'une centaine de tickets dont l'acquisition a été faite par différents terminaux mobiles. Ces images correspondent également à des tickets de bonne qualité visuelle, comme on peut le voir sur la Fig : 2.12b, même si nous avons parfois introduit quelques perturbations afin de tester notre processus d'acquisition.

À ces deux bases ont été associées des informations de « vérité terrain ». D'une part, sur chaque ticket des deux bases, nous avons construit « à la main » le rectangle englobant au mieux le ticket. D'autre part, sur 20 tickets de la base B2, nous avons extrait les informations de texte à lire automatiquement.

2.3.2. Acquisition Guidée de l'image

La première étape est l'accompagnement lors de l'acquisition de l'image du ticket de caisse. Le but de cet accompagnement est d'assurer une qualité d'image suffisante afin de faciliter la suite du traitement. Rappelons également qu'il est demandé à l'utilisateur de choisir un fond uni plutôt foncé.

L'accompagnement est basé sur la détection automatique et en temps réel des bords du ticket. Cette détection fait intervenir 7 paramètres (voir section 2.2.4.b). Pour ce test, nous avons donc utilisé la base B2 puisqu'elle correspond aux acquisitions avec smartphone. Nous avons alors procédé à une large exploration des valeurs possibles pour ces 7 paramètres. Pour évaluer la qualité de la localisation du ticket nous avons utilisé la métrique IoU : « Intersection over Union » et nous avons arbitrairement considéré qu'un ticket était correctement localisé si l'IoU était supérieur à 0.7 (dans le cas où le ticket localisé et le ticket de référence ont des surfaces similaires, cette valeur correspond à une surface commune de l'ordre de 82%). Nous avons ainsi obtenu 96% de bonne localisation avec le jeu de paramètres suivant :

$$\textit{Composante bleue},\ M = 9\ ,\ N = 20\ ,\ x = 3\ ,\ \Delta_1 = 20\ ,\ \Delta_2 = 40 \text{ et } Th = 35$$

Ces performances sont d'autant plus satisfaisantes que celles-ci sont mesurées sur des images statiques alors que l'algorithme est prévu pour être utilisé en temps réel avec un contrôle humain pour s'assurer que l'acquisition est bien réalisée, c'est-à-dire que le ticket coïncide bien avec le cadre détecté.

Les erreurs de localisation correspondent à des variations de luminosité (présence d'une ombre ou au contraire d'un reflet lumineux sur ou autour du ticket) ou à la présence d'objets à côté du ticket lors de la prise de vue. Dans de telles conditions, le détecteur risque de confondre les bords de l'objet en question avec ceux du ticket. Les fausses détections peuvent être également dues à un fond légèrement texturé et peu contrasté par rapport au ticket, comme on peut le voir sur la Fig.2.13.

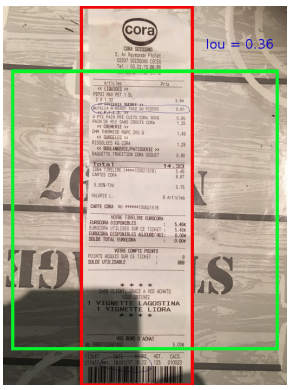


FIGURE 2.13. – Exemple de mauvaise détection (en vert) sur une image dont la couleur du fond n'est pas assez contrastée par rapport à celle du ticket et où le fond n'est pas uni, IoU = 0,36.

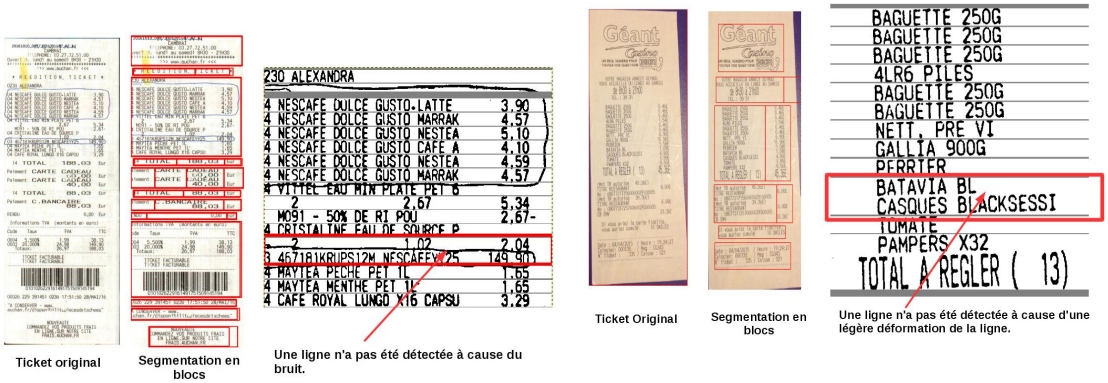
Même si les performances obtenues sont satisfaisantes, la contrainte d'acquisition imposée à l'utilisateur a été jugée trop lourde par la société AboutGoods et elle ne pourra pas être imposée systématiquement. L'alternative est de laisser l'acquisition totalement libre. La détection du ticket devra alors se faire après l'acquisition, côté serveur. Il paraît évident que le détecteur de tickets spécifique que nous avons développé ne sera plus suffisant. En effet tous les forts a priori sur lesquels repose ce détecteur ne seront certainement plus respectés dans une acquisition libre. Il sera donc nécessaire de développer un détecteur de ticket plus robuste. Le choix de laisser l'acquisition libre évitera de perdre des utilisateurs refusant des contraintes d'acquisition, mais entraînera probablement des situations où l'image acquise ne sera pas exploitable.

2.3.3. Prétraitement et Localisation des lignes de textes

Binarisation et nettoyage de l'image : Ces opérations ont pour but de faciliter la localisation des zones de textes. Plusieurs paramètres sont à prendre en compte (voir sec. 2.2.5.a) : la taille de la zone de voisinage (« k »), la valeur de la constante soustraite à la moyenne gaussienne (« c ») et la taille du filtre médian éliminant le bruit de binarisation (« m »). Pour sélectionner le meilleur jeu de paramètres, nous avons exploré systématiquement un large espace de valeurs possibles : $k \in [51, 201]$ avec un pas de 10, $c \in [9, 53]$ avec un pas de 4 ; $m \in [3, 15]$ avec un pas de 2. Le critère de qualité utilisé est la performance de la segmentation du ticket en lignes qui suit la binarisation. Ces expérimentations, effectuées sur les bases B1 et B2, nous ont conduits à choisir $k = 101$, $c = 13$ et $m = 3$.
À noter que le résultat de la binarisation est fortement dégradé lorsque le ticket est froissé, comme illustré sur la Fig.2.14. Les pliures du ticket donnent naissance à des pixels noirs qui n'appartiennent pas à des caractères.

FIGURE 2.14. – Les limites de la binarisation lorsque le ticket est froissé.

Localisation des lignes de texte Comme nous l'avons justifié dans la section 2.2.6, nous avons avons fait le choix de segmenter les tickets en lignes en utilisant une approche itérative par projections horizontales et verticales isolant les blocs de texte, puis les lignes. Les tests ont été effectués sur les deux bases B1 et B2.
Pour la première base de test, où les tickets ont été scannés, la segmentation par la méthode des projections donne de très bons résultats. En effet, plus de 95% des lignes des tickets de caisse sont correctement segmentées. Une ligne correctement segmentée est une ligne contenant une unique ligne de texte entièrement visible (pas de perte d'information sur la partie inférieure ou supérieure des caractères). Les erreurs rencontrées sont généralement dues aux informations ajoutées sur le ticket comme les zones de tickets entourées manuellement au stylo (exemple de la figure 2.15a où la présence d'un trait de stylo perturbe la segmentation). La très mauvaise résolution de certaines images peut également être une source d'erreur.
Quant à la deuxième base de test où les tickets ont été acquis par un terminal mobile, les performances descendent à 86% de lignes bien segmentées. Malgré un ensemble de tickets propres ayant subi très peu de dégradations, la méthode est sensible aux déformations du ticket qui se traduisent par de légères ondulations des lignes comme illustrées sur la figure 2.12b.



(a) Segmentation en ligne d'images scannées et situation type d'erreur.

(b) Segmentation en lignes de tickets acquis par un smartphone avec le détecteur spécifique de contours.

FIGURE 2.15. – Résultats de la segmentation en lignes par la méthode de projection sur les tickets caisses des deux différentes bases de tests.

En conclusion, l'étape de détection du texte est globalement satisfaisante, mais doit également gagner en robustesse, en particulier si l'acquisition est libre.

OCR	Tesseract	ABBYY FineReader	Google Vision
Moyenne par ticket des distances de Levenshtein	48	44	32
% de lignes lues sans erreur parmi les lignes correctement segmentées	87	87	94
% les lignes correctement segmentées	96		

TABLE 2.1. – Performance de lecture sur 20 tickets représentant 15 000 caractères.

2.3.4. Lecture des caractères

Comparaison des OCR : Nous avons donc testé trois OCR du commerce : Tesseract, ABBYY FineReader et Google Vision. Les tests ont été effectués sur un sous-ensemble de 20 tickets de la base B2 sur lesquels nous avons construit une vérité terrain du texte à reconnaître, l'ensemble correspondant à peu près à 15000 caractères et 765 lignes. Les images fournies aux trois OCR sont des images en niveaux de gris correspondant aux résultats de la segmentation en lignes. Afin de mesurer la différence de performance entre les trois OCRs testés, nous avons utilisé la distance de Levenshtein [23] entre le texte reconnu sur une image de ligne de ticket et le texte réel. Il est en effet difficile de donner une mesure de performance sur les caractères, car le fonctionnement de ces OCR n'est pas complètement connu, et certains utilisent des dictionnaires afin de corriger des erreurs de lecture. Les résultats reportés sur le Tab.2.1 donnent d'une part les distances de Levenshtein moyennes par ticket, distances mesurées sur les lignes fournies aux OCR, et d'autre part le pourcentage de lignes complètement lues (distance de Levenshtein nulle) parmi les lignes correctement segmentées. Cela permet une comparaison relative faisant clairement apparaître que Google Vision est sensiblement plus performant que les deux autres OCR, et qu'ABBYY FineReader est un peu plus performant que Tesseract.

Influence de l'orientation de l'image : L'image du document n'est pas toujours exactement verticale et l'horizontalité des lignes n'est pas non plus parfaite. Cela peut provenir d'une acquisition de l'image mal maîtrisée ou de la mauvaise conservation du ticket (froissé, plié...). Ces décalages d'orientation suffisent parfois à perturber la reconnaissance de caractères.

Un exemple d'une telle situation est proposé dans le tableau 2.2. La segmentation en ligne n'ayant pas pu fonctionner du fait de la non-horizontalité des lignes, c'est l'image des quatre lignes qui a été fournie aux OCR. On peut constater que les OCR testés, qui ont fait leurs preuves sur des documents propres, sont très sensibles à l'inclinaison des lignes. Google Vision, le plus récent d'entre-eux, est celui qui s'en sort le mieux, mais beaucoup de caractères sont omis et l'organisation en lignes est même parfois modifiée.

Tesseract	ABBY FineReader	Vision Google
ASINU GEANTST "2'3§§"Mggaégscvwaéacwmea, DU umm Au sAMEnI DE 04 76 15 04 00	VOTRE MAGASIN CASINO M(uu |"nD1 Ail SAMEDI DE «0 A 2IH30 04 76 15 04 00	vel NMAGASIN CASINO GEANT ST MARTIN HERES vOUS DU LUNDI AU SAHEDYDE 3H30 A 21H30 04 76 15 04 00

TABLE 2.2. – Influence de l'inclinaison du texte sur la performance des différents OCRs sur le marché actuel.

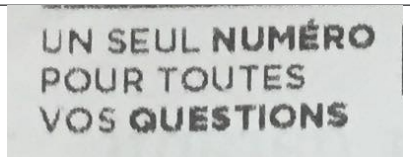
Présence de graphismes différents : Il est également intéressant de constater que la présence d'autres informations que du texte standard peut venir perturber les performances de l'OCR. Comme on peut le voir dans le tableau 2.3, nous avons testé le résultat de chaque OCR sur une même image avec ou sans textes au graphisme particulier. Il apparaît très clairement que Tesseract fait des erreurs de lecture sur des caractères qu'il lit correctement sans les graphismes perturbateurs. Pour Google Vision la perturbation se traduit par des non-détections. Seul ABBY FineReader continue à déchiffrer les caractères malgré les graphismes supplémentaires.

Influence de l'orientation du ticket : Sur nos deux bases de tests, les trois OCR ont tous été perturbés par les images de ticket tournées de 90 ° ou de 180 ° . La rotation de 90 ° est aisée à détecter, un ticket étant généralement plus long que large. La rotation de 180 ° est moins simple à détecter et cette situation devra être gérée dans le système final.

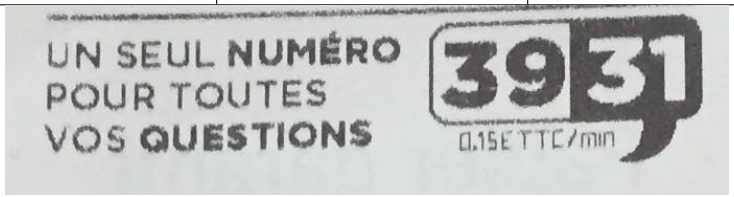
Bilan : Ces différentes analyses amènent les conclusions suivantes :

— un certain nombre de situations ont mis les OCR en défaut, ce qui justifie l'importance des prétraitements en amont et en particulier la nécessité d'une localisation propre des zones de texte ;

— en termes de performances, globalement Google Vision l'emporte sur les deux autres OCR. Cependant, utilisable sous forme d'API, c'est une boîte noire avec peu de paramètres accessibles. Le temps d'exécution est systématiquement le même, environ 5 secondes, que l'image soit très petite, comme une ligne de texte par exemple, ou très grande comme un ticket de caisse en entier. Et dans le cas d'une grande image, beaucoup de zones ne sont pas traitées. À cela s'ajoute le caractère payant de chaque requête faite à Google Vision, ce qui ajoute une contrainte très forte sur le plan économique pour le système envisagé.



Tesseract	ABBY FineReader	Vision Google
UN SEUL NUMERO POUR TOUTES VQS QUESTIONS	UN SEUL NUMÉRO POUR TOUTES VOS QUESTIONS	UN SEUL NUMERO POUR TOUTES VOS QUESTIONS



Tesseract	ABBY FineReader	Vision Google
*WWW_LJN SELIL NUMERO POUR TOUTES VOS UmYIC/rml'x	UN SEUL NUMÉRO POUR TOUTES ml8n VOS QUESTIONS 0.15 f ÎÎC/mm	3931 POUR TOUTES VOS QUESTIONS TT/min

TABLE 2.3. – Influence de la présence de graphisme autour du texte sur la performance des différents OCRs testés.

Pour ces raisons, nous avons donc fait le choix d'utiliser Tesseract dont la souplesse d'utilisation et les performances globalement satisfaisantes nous semblent correspondre à notre besoin.

2.3.5. Analyse Sémantique

Comme nous l'avons dit précédemment, l'analyse sémantique a deux rôles principaux. Elle permet dans un premier temps de corriger les erreurs de l'OCR et, dans un deuxième temps, d'extraire les informations contenues dans le ticket de caisse. Cette analyse est effectuée sur les mots fournis par l'OCR.

La correction des erreurs est effectuée en utilisant simplement un dictionnaire.

En ce qui concerne l'extraction d'informations, pour ce démonstrateur, nous nous sommes limitée à l'extraction d'informations dites « de bases » telles que l'enseigne, la date d'achat, le nombre d'articles, le montant total, etc. Nous ne cherchons pas à interpréter les libellés courts. Les performances de l'extraction sont naturellement fortement liées aux performances de l'OCR. Sur une image parfaitement retranscrite par l'OCR, les performances sont très bonnes, reportées sur la Tab. 2.4. L'objectif étant de mesurer les performances de l'analyse sémantique, nous ne prenons pas en compte les quelques erreurs de l'OCR (par exemple si il y a une erreur au niveau d'un chiffre dans le montant total).

Enseigne	Ville/Code Postal	Date et Heure	Montant Total	Nb Articles
92%	78%	90%	84%	79%

TABLE 2.4. – Performances Analyse Sémantique sur la base B1.

Les erreurs se situent essentiellement sur des informations ambiguës dans le ticket. Par exemple, l'information « date d'achat » peut-être confondue avec d'autres dates présentes sur le ticket, comme la date à partir de laquelle les points fidélité sont disponibles. L'information « ville » peut être confondue avec certains produits tels que « Perrier », « Orange » ou encore « Evian ». Où encore certains tickets dans lesquels figurent plusieurs montants différents, le montant par catégorie de produits, le montant avant remises, le montant réel à payer, etc.

À ce stade, les solutions pour remédier aux différentes situations ambiguës citées ci-dessus ne sont pas directes. Une des solutions serait de pouvoir délimiter les différentes zones de textes, en identifiant la structure du ticket. On peut alors parler de segmentation sémantique du ticket. Cette solution permettrait une segmentation plus précise, en localisant non seulement la zone de libellés produits, mais aussi le logo, l'entête, le bas du ticket et d'autres graphismes tels que les code barres.

2.3.6. Verrous technologiques

Le bilan de cette première analyse du problème de la lecture de ticket de caisse fait apparaître deux éléments fondamentaux. D'abord, il n'existe pas actuellement de solution totalement satisfaisante adaptée à la lecture automatique de tickets de caisse à partir d'une image prise par un smartphone et permettant d'atteindre les performances attendues par AboutGoods. Ensuite, le problème fait apparaître une certaine complexité, en grande partie liée à la volonté de pouvoir disposer d'un système robuste capable de prendre en compte des situations délicates où l'image transmise par le smartphone n'est pas de bonne qualité. Les points qui nous semblent être les plus critiques sont :

1. la localisation précise du ticket dans l'image qui doit permettre de supprimer l'imposition de contraintes lors de l'acquisition et ne transmettre que l'image du ticket au module d'analyse. Il faudra également que cette étape assure que le ticket soit dans le bon sens de lecture et qu'il soit redressé. Comme on a pu le voir, l'OCR est très sensible à l'inclinaison et à la déformation du ticket ;
2. l'analyse de la structure du ticket. L'objectif est de pouvoir localiser et catégoriser les différentes zones du ticket de caisse à savoir le logo, le code barre, et les zones de texte. L'objectif est de sélectionner précisément les zones de texte pour maximiser la performance de l'OCR ;
3. la classification du logo et la reconnaissance de l'enseigne. La détermination de

l'enseigne permettrait d'apporter des connaissances a priori sur la structure du ticket et faciliter ainsi son analyse ;

4. l'extraction des informations sur le ticket et surtout l'interprétation des libellés courts, peu explicites, sans ambiguïté et ce malgré les erreurs possibles de l'OCR. Par exemple de pouvoir automatiser l'analyse d'un libellé court tel que : « pdm sdw ceral » en « pain de mie sandwich aux céréales ».

2.3.7. Apports du Deep Learning

Ces dernières années l'apprentissage profond ou Deep Learning a permis d'obtenir des résultats inédits dans tout un ensemble de domaines, dont la compréhension automatique des images, surpassant bien souvent les méthodes de l'état de l'art. Dans la construction d'une architecture Deep Learning, la phase d'apprentissage est souvent longue et coûteuse et peut requérir une importante base d'apprentissage, mais les architectures résultantes peuvent rester légères et convenir à notre cadre applicatif. Aussi, il nous semble incontournable d'introduire dans notre système des solutions basées sur des réseaux profonds.

Comme nous le verrons plus en détail dans le chapitre suivant, Le Deep Learning est déjà très présent dans l'analyse de documents dématérialisés et pour les outils de reconnaissance optique de caractères [24, 25, 21, 26]. Les auteurs du papier « Deep Features for Text Spotting » [25] ont entraîné un réseau de neurones permettant de réaliser à la fois la localisation et la reconnaissance des caractères sur des images de scènes naturelles. Des performances de 98% pour la localisation des zones de textes et de 91% pour la classification des caractères sont alors possibles sur le jeu de données considéré.

Dans les travaux de cette thèse, le Deep Learning interviendra à différentes étapes détaillées dans la section suivante 2.4.

2.4. Chaîne de traitement envisagée

En fonction de l'expérience acquise à travers la construction du démonstrateur et compte-tenu des fortes contraintes industrielles imposées, en particulier liées à la fiabilité de chaque information extraite (voir section 2.2.2), nous proposons la chaîne de traitement décrite sur la figure 2.16. Nous allons maintenant décrire les différents éléments composant cette chaîne, mais il est important de noter que la plupart des éléments proposés sont constitués de deux sous-éléments complémentaires afin de renforcer la fiabilité du système.

La chaîne de traitement proposée comprend un certain nombre d'étapes. Dans les sections précédentes, nous avons souligné l'importance de la préparation, du prétraitement de l'image afin d'assurer une lecture optimale du ticket de caisse. C'est pourquoi les trois premières étapes correspondent à une phase de prétraitement préparant la phase

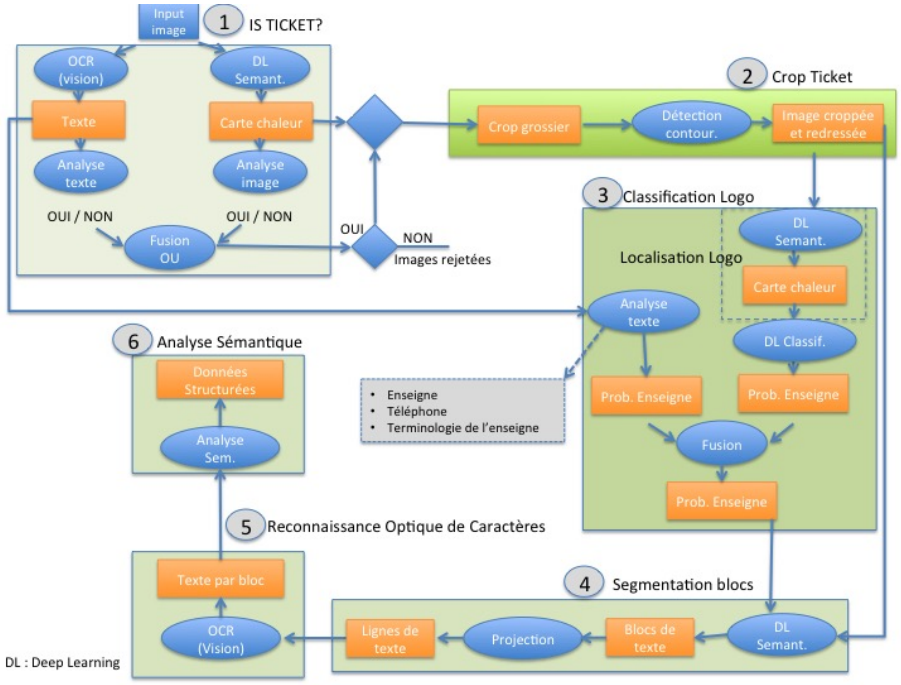


FIGURE 2.16. – Chaîne complète du système de lecture automatique de tickets de caisse

de lecture.

La première étape a pour objectif la vérification de la présence d'un ticket dans l'image. En effet, dans les applications qui offrent la possibilité de prendre des photos, il n'est pas rare que les utilisateurs ne suivent pas les règles et envoient des images floues, sans informations utiles, des selfies, etc. Il est évident qu'essayer de traiter des images qui ne contiennent pas de ticket de caisse ne ferait que polluer l'ensemble du système et il est donc important de filtrer ces images. Pour cela, nous proposons de fusionner les résultats de deux analyses menées en parallèle. Une première analyse utilise un premier réseau profond effectuant une segmentation sémantique des pixels, les classes possibles pour un pixel étant « ticket » ou « non ticket ». La deuxième analyse utilise un OCR du commerce appliqué sur toute l'image du ticket. Sans segmentation préalable, le résultat est très incomplet bien évidemment, mais il suffit de détecter certains éléments correspondant à un ticket de caisse pour valider la détection. Par exemple, la lecture d'une ligne produit est une forte présomption de la présence d'un ticket de caisse. La décision finale est prise en fusionnant les deux résultats, le but étant d'assurer qu'aucune image contenant un ticket de caisse exploitable ne soit écartée.

La deuxième étape consiste à localiser de façon précise le ticket de caisse afin d'obtenir une image détourée du ticket autrement dit une image débarrassée de son contexte d'acquisition. Cette étape permettra de faciliter la détection de texte. Cette étape est réalisée

en deux temps. Dans un premier temps, le résultat de la segmentation sémantique de l'étape précédente permet d'avoir une première estimation, grossière, de la localisation du ticket. Dans un deuxième temps, le détecteur de contours utilisé dans le démonstrateur affine la position du ticket et procède éventuellement à son redressement s'il n'est pas parfaitement vertical.

La troisième étape est la détection de l'enseigne. Cette étape a pour but d'apporter un maximum de connaissance a priori sur la structure du ticket avant de commencer sa lecture. Chaque enseigne a une manière spécifique de présenter un ticket de caisse, des polices et tailles de police propres, des espacements particuliers entre lignes, entre caractères... Comme pour la première étape, nous allons assurer la détection de l'enseigne en parallélisant deux analyses. Une nouvelle lecture du ticket de caisse par un OCR sur l'image globale détourée (l'OCR sera forcément un peu plus performant qu'à la première étape de détection du ticket, car il n'est pas perturbé par le fond de l'image, mais il ne sera cependant pas suffisamment performant pour terminer l'analyse), permet d'extraire des informations pouvant éventuellement donner un indice sur l'enseigne (pour certaines enseignes, le logo comprend le nom de l'enseigne). En parallèle une analyse de l'image à base de réseaux profonds cherche à localiser puis à identifier le logo. Une règle de fusion entre ces deux approches permet de prendre une décision fiable. À noter que la détection du logo a été construite pour être capable de détecter les tickets retournés (haut à la place du bas) de manière à pouvoir fournir à l'OCR des éléments correctement orientés.

À la sortie de l'étape 3, nous disposons d'une image limitée au seul ticket, redressée, si nécessaire, et nous avons connaissance de l'enseigne. Dans une quatrième étape, nous nous intéressons à la segmentation en blocs de zones de texte. Cette étape enchaîne deux opérations. La première opération, effectuée avec un réseau profond, réalise une segmentation sémantique du ticket en blocs afin de séparer l'entête, le logo, la liste de produits, la liste les prix, etc. L'intérêt est double. D'une part, cela permet de préparer la segmentation en ligne qui va suivre, et d'autre part cela permet également de faciliter de travail de correction qui interviendra plus tard. Par exemple, si dans la liste des prix l'OCR a reconnu « 4O », la correction de la lettre « O » par le chiffre « 0 » est naturelle. La seconde opération de cette étape est la segmentation en lignes effectuée par projections horizontales.

Ensuite, dans une cinquième étape, chaque bloc est soumis à un OCR .

Dans une dernière étape, nous réalisons une analyse sémantique adaptée en fonction de la nature des blocs.

2.5. Synthèse et conclusion

Ce chapitre a permis de mettre en évidence les objectifs et le contexte industriel du travail envisagé. Après une veille technologique sur les systèmes de lecture de tickets de

caisse existants, nous nous sommes intéressée à la réalisation d'un premier démonstrateur en exploitant des techniques simples d'analyse d'images de documents. Cela a permis de démontrer la faisabilité de l'application et mettre en évidence les verrous technologiques à surmonter. Enfin pour répondre aux contraintes imposées par le contexte industriel (fiabilité des résultats, ergonomie de l'application) et, suite à l'évaluation du démonstrateur, nous avons pu proposer une chaîne de traitement complète comportant six étapes.

3. État de l'art

Dans ce chapitre, nous présentons l'état de l'art sur les différentes thématiques liées au problème abordé. Nous commençons par décrire, de manière générale, les différentes approches basées sur le Deep Learning pour la classification d'images et la détection d'objets. Puis nous examinons plus précisément, les techniques adaptées aux différents objets que nous souhaitons localiser, à savoir le ticket de caisse, le logo, les zones de textes. Enfin, nous terminons par présenter les différentes techniques d'analyse sémantique qui nous intéresseront dans nos réalisations.

3.1. Introduction

Précisons d'abord un peu de vocabulaire. De manière conventionnelle, la classification d'images, ou de sous-images, consiste à attribuer une classe à une image, ou à une sous-image, parmi un ensemble de classes déterminées. Et nous appellerons « détection d'objet » le processus consistant à la fois à localiser un objet dans une image (par un rectangle englobant ou par un contour fermé) et à attribuer une classe à cet objet. En ce sens, la détection d'objet est plus précise que la classification d'images, ou de sous-image.

Nous avons vu au chapitre 2 que peu de travaux traitent de l'analyse des tickets de caisse. Néanmoins, à travers l'analyse de ces quelques travaux et nos propres expérimentations, nous avons pu mettre en évidence trois étapes principales dans la lecture automatique d'un ticket de caisse :

— la détection des éléments qui vont permettre la lecture du ticket, à savoir le ticket lui-même, le logo puis les zones de texte, ces zones pouvant être des blocs, des lignes ou des caractères ;
— la reconnaissance de caractères appliquée aux zones de texte identifiées à l'étape précédente ;
— l'analyse sémantique permettant une compréhension des caractères lus.

La première de ces étapes peut être assimilée à des problématiques plus générales de classification d'images (cette image est-elle une image de ticket ?) et de détection d'objets dans une image (les objets étant ici un ticket, un logo et des zones de texte). Une première section de ce chapitre sera donc consacrée, de manière assez générale, à la classification d'images et à la détection d'objets (section 3.2), en se focalisant plus particulièrement sur les méthodologies s'appuyant sur le « deep learning ». En ce qui concerne les logos et les zones de texte, il existe un certain nombre de travaux spécifiquement dédiés à ces tâches. Nous examinerons ces travaux dans les sections 3.3 et 3.4. La sous-section 3.4.3 sera consacrée à la reconnaissance de caractères, même si nous verrons que souvent localisation et reconnaissance du texte sont parfois étroitement mêlées. Enfin, dans la dernière section 5.5, nous évoquerons rapidement l'état de l'art sur l'analyse sémantique.

3.2. Le Deep Learning pour la classification d'images et la détection d'objets

3.2.1. Introduction

La classification d'images, et plus encore la détection d'objets, jouent des rôles très importants dans l'analyse d'images, car elles sont à la base de la compréhension du contenu d'une image. Depuis bientôt un demi-siècle, ces thématiques ont fortement mobilisé la communauté des chercheurs en image, et une multitude d'approches ont été proposées.

Une manière classique de réaliser une classification d'images consiste à procéder en deux étapes principales. Dans un premier temps, des caractéristiques, globales ou locales, sont extraites de l'image, ces caractéristiques devant être compactes et pertinentes vis-à-vis de la classification envisagée. Dans un deuxième temps, une classification est effectuée sur ces caractéristiques. Selon le problème et l'information disponible a priori, cette classification peut être supervisée ou non. La multitude de travaux de la littérature montre une grande variété à la fois dans les caractéristiques utilisées et les méthodes de classification [27].

En ce qui concerne la détection d'objets, les premiers travaux (années 1980) reposaient plutôt sur des approches cherchant à retrouver, dans les images de contours, des modèles géométriques [28]. À partir des années 1990, ce sont les modèles d'apparence qui ont plutôt été utilisés, apportant plus de robustesse à la variabilité à l'intérieur d'une même classe d'objets. En particulier, les années 2000 ont vu le développement des approches dites par « sacs de mots visuels » [29].

Plus récemment, aussi bien pour la classification d'images que pour la détection d'objets, les approches par « Deep Learning », par leur capacité à apprendre automatiquement des modèles suffisamment génériques, ont bouleversé le panorama, supplantant par leurs performances la plupart des autres approches. C'est donc vers ces approches que nous nous sommes plutôt orientée, et c'est pourquoi nous commençons cet état de l'art par une présentation des approches Deep Learning dédiées à la classification d'images et la détection d'objets.

3.2.2. Deep Learning : introduction

Une architecture « Deep Learning » est une architecture à base de réseaux de neurones artificiels comportant un grand nombre de couches cachées. Ces architectures s'inspirent des perceptrons multicouches à propagation directe et à rétropropagation du gradient proposés par Rumelhart en 1986 [30]. Dans ces architectures à plusieurs couches, chaque neurone d'une couche est connecté à l'ensemble des neurones de la couche précédente. La rétropropagation du gradient permet, grâce à une base d'apprentissage annotée et par un processus itératif progressant de façon inverse, l'ajustement des paramètres du réseau. Cependant, dans le cas d'images de grande taille, ces réseaux dits « totalement connectés » aboutissent à des réseaux trop complexes comportant beaucoup trop de paramètres.

Aussi, dans le cas de l'analyse d'images, les architectures profondes proposées sont

souvent à base de réseaux convolutionnels (CNN pour *Convolutional Neural Networks*). Dans les réseaux de neurones convolutionnels, les premières couches consistent généralement en une succession de couches de convolution (nous détaillerons ces couches dans la section suivante) agissant comme des extracteurs de l'information pertinente contenue dans les images, tandis que les couches finales sont des couches totalement connectées (comme pour les perceptrons multicouches) et sont dédiées à la classification. L'utilisation d'un grand nombre de couches de convolution permet une croissance progressive du niveau d'abstraction des informations extraites, ce qui permet de découvrir des indicateurs de « haut niveau » générant un fort pouvoir prédictif, prédiction réalisée par la seconde partie du réseau composé de couches complètement connectées. Ces réseaux permettent d'aboutir à des architectures avec moins de paramètres que les réseaux totalement connectés (une même convolution étant partagée par l'ensemble des pixels). Ils apportent également une invariance à la translation intéressante pour la détection des objets dans les images. Ces architectures ont atteint des performances inédites dans un certain nombre de tâches en vision par ordinateur notamment dans les domaines de la reconnaissance d'images, avec une publication clé [31] lors de la conférence NIPS'2012.

Généralement, même s'ils sont moins complexes que les réseaux complètement connectés, ces réseaux comportent encore un très grand nombre de paramètres, et leur entraînement, toujours par rétropropagation du gradient, requiert la disponibilité de grandes bases de données annotées et des moyens de calcul importants. Ceci est rendu possible avec l'arrivée du Big Data et par la disposition de processeurs graphiques (GPU).

Nous allons présenter maintenant de manière un peu plus détaillée les caractéristiques des réseaux de neurones convolutionnels.

3.2.3. Architecture des réseaux de neurones convolutionnels

Un réseau de neurones convolutionnel fait apparaître trois principaux types d'éléments : des couches de convolutions, des opérateurs d'agrégation et des couches complètement connectées. Classiquement, l'architecture d'un tel réseau comporte d'abord un enchaînement de plusieurs ensembles « couche de convolution + opérateur d'agrégation » et se termine par des couches complètement connectées.

Couches de convolution Une couche de convolution est constituée d'un ensemble de N opérations de convolutions, chacune caractérisée par son masque de convolution de taille $K_x \times K_y \times K_p$, K_x et K_y désignant les dimensions spatiales du masque et K_p la profondeur du masque, c'est-à-dire le nombre de composantes de l'image en entrée de la couche de convolution. Ces convolutions sont généralement suivies d'une fonction non-linéaire, appelée fonction d'activation. La fonction d'activation peut prendre différentes formes. Une solution très courante est l'utilisation de la fonction ReLU (Unité Linéaire Rectifiée) proposée par [32], dont l'expression est $f(x) = max(0, x)$. Cette fonction a la particularité d'être non saturante contrairement à d'autres fonctions d'activation classiques comme $f(x) = tanh(x)$ où $f(x) = (1 + \exp^{-x})^{-1}$. Dans [31], il est montré que la simplicité de cet opérateur permet d'accélérer de façon non négligeable la phase d'en-

traînement.
La sortie d'une couche de convolution est alors un ensemble de N images, appelées « cartes d'activation ». En fonction de la taille spatiale des masques, le pas de déplacement spatial d'un masque de convolution (*stride*) peut ne pas être de 1, ce qui permet de réduire la taille des images de sortie. La gestion des bords des images lors de la convolution peut également être prise en compte par agrandissement spatial de l'image, typiquement avec des valeurs nulles (*padding*).

Agrégation (« pooling ») Dans les architectures traditionnelles, les couches de convolution sont souvent suivies par des couches d'agrégation (« pooling »), permettant un sous-échantillonnage spatial (donc une réduction de la taille des données générées) en même temps qu'une agrégation de l'information. Le « pooling » apporte également une invariance aux faibles rotations et translations [33]. Il existe différentes solutions de pooling, la plus répandue étant le « max-pooling » ou « max-agrégation », consistant à ne conserver que la valeur maximale à l'intérieur d'une fenêtre de taille donnée (typiquement 2×2 ou 3×3).

Couches de neurones complètement connectés Les couches de convolution sont supposées avoir extrait une représentation compacte et discriminante de l'information contenue dans l'image. Mise sous forme de vecteur, cette information est alors appliquée en entrée d'un ensemble de couches complètement connectées, retrouvant la forme classique d'un réseau de neurones où chaque composante du vecteur d'entrée est connectée à tous les neurones de la couche. Notons que cette connexion totale induit généralement un nombre de paramètres très important. L'objectif de ces couches est de réaliser la classification recherchée. En pratique, quelques couches complètement connectées sont ajoutées au réseau, la dernière couche, dans le cas d'un apprentissage supervisé, contenant autant de neurones que de classes désirées. Une fonction d'activation de type « SoftMax », dont l'équation est donnée ci-dessous, est souvent utilisée afin d'obtenir des probabilités d'appartenance à chaque classe. Si $Z = (z_1, ..., z_K)$ est la sortie de la dernière couche dans un problème de classification à K classes, cette sortie est transformée en un vecteur de probabilités $\sigma(Z)$ tel que :

$$\sigma(z_k) = \frac{e^{z_k}}{\sum_{k=1}^{K} e^{z_k}} \quad k \in 1, ..., K \tag{3.1}$$

Le choix de l'architecture d'un réseau convolutionnel profond demande donc de déterminer un grand nombre d'éléments : nombre de couches de convolution, nombre de convolutions par couche, taille des masques de convolution, forme de la fonction d'activation, forme du pooling, nombre de couches totalement connectées et nombre de neurones pour chacune de ces couches... Ces choix restent pour le moment en grande partie empiriques.

3.2.4. Entraînement d'un réseau de neurones

La phase d'apprentissage d'un réseau de neurones consiste, à partir d'une base de données annotée, à optimiser les nombreux paramètres du réseau afin de minimiser le

taux d'erreur de prédiction du réseau. Les réseaux de neurones profonds, en général, sont entraînés en utilisant un algorithme de rétropropagation dit de « descente du gradient ». Il s'agit en fait de minimiser une fonction coût construite sur l'écart entre la prédiction et le modèle, et ce par itérations successives en se basant sur les dérivées de la fonction coût par rapport aux paramètres du réseau. Une phase de test peut être introduite en parallèle à l'apprentissage afin d'évaluer les performances du réseau sur des données différentes de celles de la phase d'apprentissage.
Pour être en mesure de réaliser un apprentissage robuste et convergeant, il est important de veiller à un certain nombre de paramètres contrôlant, dans la pratique, la manière dont les poids des neurones vont être optimisés. Ces hyperparamètres (« epoch », « batch », « itération », « learning rate ») ont une importance fondamentale sur la rapidité et la qualité de l'apprentissage. Une « epoch » correspond à un unique passage aller/retour de l'ensemble des données d'apprentissage. Comme une « epoch » est généralement trop grande pour alimenter le réseau en une seule fois, celle-ci est divisée en plus petits lots appelés « batch ». Une itération est le nombre de « batch » nécessaire afin de compléter une « epoch ». Le « learning rate » est le paramètre qui gère le taux de mise à jour des paramètres entre deux itérations. Ce taux peut être fixe, ou évoluer au cours de l'apprentissage. Un autre paramètre important de l'apprentissage est le « dropout ». Le principe est de désactiver un certain nombre de neurones durant la phase d'entraînement, le choix de ces neurones étant fait de manière aléatoire à travers une probabilité fixe (souvent choisie égale à 0,5). Le « dropout » est une forme de régularisation permettant de minimiser la dépendance inter-neurones et, en conséquence, de limiter le sur-apprentissage (phénomène qui consiste à « trop bien » apprendre les données d'apprentissage au détriment d'une mauvaise capacité de généralisation des modèles obtenus). A noter que l'effet du dropout est amélioré dans les architectures « Maxout » [34]. Une unité « Maxout » peut être considérée comme une nouvelle fonction d'activation consistant à choisir la valeur maximale parmi un ensemble de valeurs de différentes cartes d'activation, réalisant ainsi un pooling non plus spatial, mais sur un ensemble de sorties d'une couche du réseau.

Comme pour le choix de l'architecture d'un réseau, le choix des hyperparamètres reste en grande partie empirique.

3.2.5. Les réseaux de neurones convolutionnels qui ont rendu le deep learning populaire

Le challenge ImageNet En préambule, comme les performances des réseaux que nous allons présenter ont souvent été mesurées sur le challenge ILSVRC (ImageNet Large Scale Visual Recognition Challenge), nous allons rapidement présenter ce challenge. L'objectif d'ILSVRC est une évaluation des algorithmes de détection d'objets et de classification d'images à grande échelle. La base ImageNet utilisée contient 14 millions d'images annotées (parmi lesquelles un million possède des rectangles englobant les objets présents) correspondant à 20 000 classes. Depuis 2012, dans ce challenge, les réseaux de neurones ont supplanté les autres approches, et depuis 2015, les performances obtenues en classi-

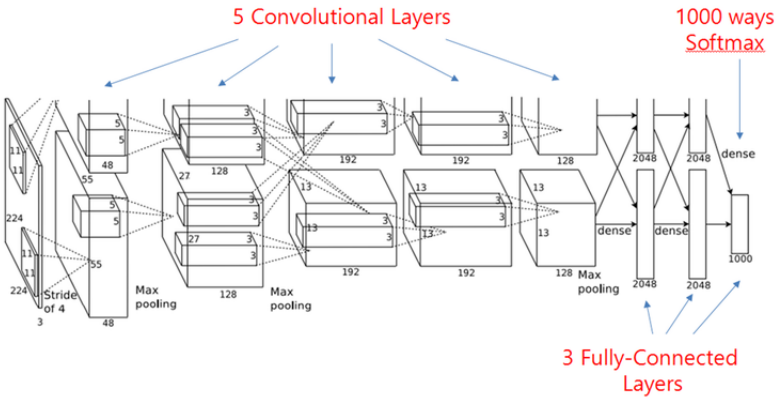


FIGURE 3.1. – Réseau Alexnet [31]

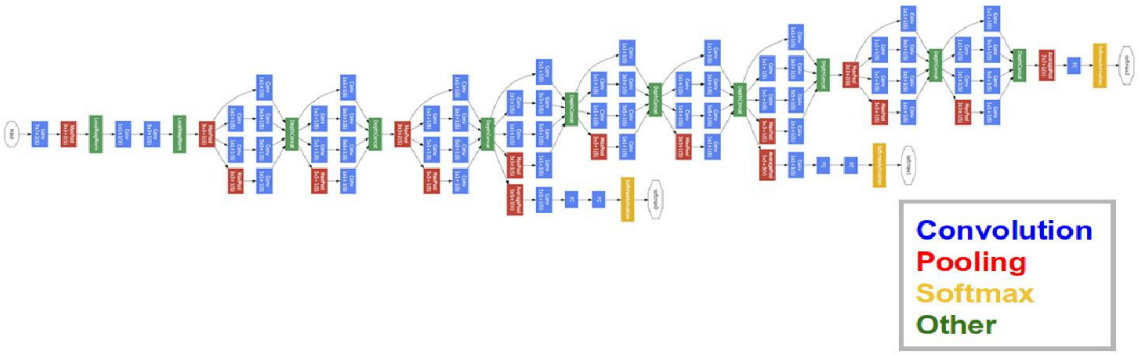
fication ont dépassé la capacité humaine.

AlexNet Le réseau avec lequel tout a commencé est le réseau AlexNet, en 2012, [31] (Fig. 3.1). L'architecture de ce réseau est relativement simple comparée aux architectures plus récentes. Le réseau est constitué de cinq couches de convolution (filtres de convolution de taille 11×11 sur les premières couches, avec maxpooling, ReLU et normalisation locale) et de trois couches entièrement connectées. Ce réseau, qui contient plus de 60 millions de paramètres, a été entraîné sur la base d'ImageNet [35] contenant, à l'époque, environ 1.2 millions d'images et 1000 catégories. Cette architecture a permis d'obtenir une erreur de 15% en classification d'images, devançant les autres équipes de plus de 10%.

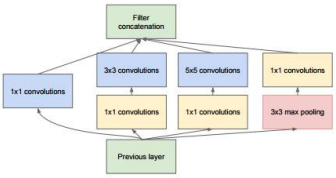
VGGNet Un autre modèle, créé en 2014, [36] nommé VGGNet, est caractérisé par sa profondeur puisqu'il comporte jusqu'à 19 couches, mais aussi par sa simplicité puisqu'il utilise des filtres de petite taille (taille maximum 3×3, très inférieure à la taille 11×11 utilisée dans d'AlexNet) et un maxpooling. L'utilisation de nombreux filtres de petite taille est préférable à l'utilisation de filtres de taille plus importante, mais moins nombreux. Ceci augmente en effet la profondeur du réseau ce qui permet d'apprendre progressivement des fonctionnalités plus complexes. Dans [36], plusieurs configurations ont été testées, de profondeur différente (de 11 à 19 couches) et en modifiant la taille des filtres de convolutions (soit 1x1, soit 3x3). La configuration atteint des performances intéressantes à partir d'architectures à 16 couches contenant exclusivement des filtres 3x3, le modèle à 19 couches étant le plus performant.

VGG atteint une précision remarquable (de l'ordre de 7%) sur l'ensemble de données ImageNet. Cependant son entraînement pose de grosses contraintes, à la fois en termes de mémoire et de temps de calcul.

GoogLeNet Egalement en 2014, Google propose une nouvelle architecture, GoogLeNet avec l'introduction du « module d'Inception » [37]. GoogLeNet contient 22 couches et



Architecture complète du réseau GoogLeNet



Module d'inception

FIGURE 3.2. – Réseau GoogLeNet , 2014 [37]

c'est l'une des premières architectures qui s'écarte de l'idée générale de l'empilement de couches de convolution et de max-agrégation les unes sur les autres dans une simple structure séquentielle.

En regardant l'architecture GoogLeNet de la Fig. 3.2, on remarque que certains blocs introduisent des opérations en parallèle et non de manière séquentielle. Ces blocs sont appelés « Module d'Inception », Fig.3.2. Un « Module d'Inception » réalise différentes opérations, typiquement des convolutions de tailles différentes, en parallèle, évitant ainsi à l'utilisateur de choisir arbitrairement une unique taille de convolution et conférant ainsi une sensibilité à différentes échelles spatiales. Les sorties de ces différentes opérations sont ensuite concaténées. L'idée « naïve » derrière ce mécanisme est de laisser le modèle « choisir » la meilleure opération grâce à la phase d'apprentissage. Cependant, pour éviter une explosion du nombre de calculs, des convolutions 1×1 sont utilisées au préalable (voir sur la Fig. 3.2), ces convolutions 1×1 étant généralement associées à une fonction d'activation non-linéaire. Ce modèle a permis d'améliorer les performances en classification et l'efficacité des calculs. À noter que pour permettre la convergence lors de l'entraînement d'un réseau aussi profond, des sorties intermédiaires de classification sont utilisées.

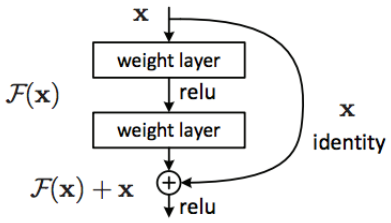


FIGURE 3.3. – Exemple de bloc Résiduel [38]

ResNet Bien que l'on ait pu constater qu'une augmentation « raisonnable » de la profondeur des réseaux aboutissait à accroître la précision de classification, toutes les architectures présentées ci-dessus ne permettent pas d'augmenter indéfiniment le nombre de couches . En effet, dans la phase d'apprentissage des poids par rétro-propagation, un trop grand nombre de couches, ne permet pas un entraînement efficace des couches bas niveau du réseau, du fait de l'atténuation du gradient. Les réseaux « résiduels » permettent la formation de réseaux très profonds en introduisant des modules appelés « modèles résiduels » comme le montre la Fig 3.3. Dans une couche de bloc résiduel, le traitement appliqué à l'entrée x est une série de convolution-relu-convolution. Au résultat obtenu, on ajoute alors l'entrée originale x, si bien que, contrairement aux CNN traditionnels, le réseau conserve une information liée à l'image originale. Ce mécanisme a pour avantage de faciliter le processus de rétro-propagation, car les opérations d'addition permettent de distribuer le gradient à travers tout le réseau et d'éviter une trop forte atténuation.

En 2015, Microsoft a proposé le réseau ResNet [38], basé sur ce principe. Très profond (152 couches), il a permis d'établir à l'époque de nouveaux records en classification. ResNet est l'une des meilleures architectures CNN qui existent actuellement.

3.2.6. Transfert d'apprentissage

L'apprentissage d'un réseau tel que décrit dans la section 3.2.4 est une phase délicate qui peut être réalisée de deux manières différentes. La première manière consiste à entraîner la totalité du réseau, c'est-à-dire à calculer chacun des paramètres du réseau durant la phase dite d'apprentissage. Ceci suppose d'une part la disposition d'une base d'apprentissage suffisamment conséquente et pertinente vis-à-vis de la tâche à réaliser, et d'autre part des moyens de calcul importants, deux conditions qui ne sont pas toujours réunies. La seconde manière consiste à réutiliser une partie d'un réseau qui a déjà fait ses preuves (par exemple l'un des réseaux présentés au paragraphe3.2.5). On parle alors d'apprentissage par transfert (*Transfer Learning*), avec ou sans ajustement fin (*Fine Tuning*).

Les termes « transfer learning » et « fine tuning » font référence à deux concepts complémentaires permettant l'adaptation à un nouveau contexte d'un réseau optimisé dans un contexte initial. Le « fine tuning » consiste à procéder à un ajustement fin de tout ou partie des paramètres déjà entraînés d'un réseau afin de les adapter au nouveau

contexte de données. Cet ajustement suppose implicitement que la distribution des nouvelles données est proche de celle des données ayant servi au réglage initial du réseau. Le « transfer learning » consiste à réutiliser une partie amont d'un réseau entraîné dans un premier contexte et de compléter l'architecture par de nouvelles couches finales qui sont alors entraînées pour répondre à la tâche du nouveau contexte. Ces deux méthodes sont complémentaires, car lors d'un transfert, la partie amont transférée peut être ajustée finement pendant la phase d'entraînement de la nouvelle partie avale du réseau. Cette stratégie permet la coadaptation des deux parties du réseau au nouveau contexte et permet d'obtenir des gains de performance significatifs [39].

De nombreux travaux ont su tirer profit de cette méthode et obtenir des résultats qui surpassent l'état de l'art dans leur domaine [40, 41, 42, 43]. Ces travaux ont utilisé comme réseau de base le réseau AlexNet [31] et considèrent que les premières couches de ce réseau de neurones calculent des caractéristiques assez générales pour être réutilisées pour répondre à un autre problème. L'article [43] montre que l'apprentissage par transfert peut-être un outil puissant pour permettre l'entraînement d'un réseau de neurones profond contenant des millions de paramètres lorsque l'ensemble de données d'apprentissage est significativement plus petit que nécessaire. Selon la taille de l'ensemble de données d'apprentissage, plusieurs solutions sont possibles :

— si l'ensemble de données est trop petit et le nombre de paramètres important, pour éviter le sur-apprentissage, il est préférable que les paramètres de la première partie du réseau restent gelés ;
— si l'ensemble de données est volumineux où le nombre de paramètres est faible (de sorte que le sur-apprentissage ne pose pas de problème), il est possible d'affiner les paramètres de la première partie pendant la phase d'entraînement, pour améliorer les performances.

Les auteurs de [43] ont pour objectif de ré-entraîner le réseau AlexNet (60 millions de paramètres) sur la base de PASCAL VOC contenant quelques milliers d'images d'entraînement (ce qui est peu relativement à la taille de ImageNet-2012 [35]). L'entraînement d'un tel réseau avec si peu de données est problématique et c'est dans ce cas que l'apprentissage par transfert prend tout son sens. Ici, les auteurs ont fait le choix de ne pas faire de « fine tuning » et de figer les paramètres de toutes les premières couches. Seule la dernière couche « fc8 » a été remplacée par deux nouvelles couches « fca » et « fcb » complètement connectées dites « couches d'adaptations ». Les résultats obtenus sont très intéressants puisqu'ils sont meilleurs que les précédents travaux sur cette même base.

Dans l'article [39], les auteurs étudient la transférabilité des caractéristiques apprises à différentes couches d'un CNN. Pour cela ils créent deux divisions de l'ensemble de données ImageNet (A / B) et explorent comment la performance varie pour divers choix de conception de réseau.

Les résultats obtenus montrent que le transfert d'apprentissage fonctionne mal lorsque la division se situe quelque part au milieu du réseau (n=3-6). Ces couches sont des

couches de convolutions et les neurones de ces couches co-adaptent les activations des neurones de la couche suivante et cette connexion complexe est brisée quand ces couches sont séparées. Les performances du transfert d'apprentissage sont nettement meilleures à mesure que l'on se rapproche des couches finales, car les fonctionnalités sont plus spécifiques à chaque tâche sur les dernières couches.

La méthode d'apprentissage par transfert avec ou sans ajustement fin sur des réseaux profonds a, comme nous avons pu le voir, de nombreux avantages et peut-être très utile pour entraîner des réseaux avec peu de données. Cette méthode peut être vue comme la capacité d'un système à reconnaître et appliquer des connaissances et des compétences, apprises à partir de tâches antérieures, sur de nouvelles tâches ou domaines partageant des similitudes.

3.2.7. Détection d'objet à l'aide du Deep Learning

Le Deep Learning, comme nous venons de le voir, a permis de faire des progrès très sensibles en classification d'image et est considéré aujourd'hui comme étant la méthode de pointe pour la classification d'images naturelles. Très rapidement, certains travaux ont été un peu plus loin en tentant de traiter la détection d'objets à l'aide de CNNs. De nombreuses approches ont été proposées dans la littérature. Nous allons présenter quelques-unes d'entre elles.

3.2.7.a. Transformation d'un réseau de classification en réseau purement convolutif

L'une des premières approches explorée est l'utilisation des réseaux de classification afin de réaliser de la localisation et de la détection d'objets. Pour cela, le réseau de classification initial est transformé en réseau purement convolutif en remplaçant les couches complètement connectées par des couches de convolutions équivalentes, afin de réaliser de la segmentation sémantique par patchs.

Afin d'expliquer cette conversion, revenons à la définition d'une couche complètement connectée à N neurones. Elle reçoit en entrée un vecteur x de taille $t = n \times n \times C$, la vectorisation d'une image de taille $n \times n$ à C canaux. Cette couche réalise l'opération $y = f(Wx + b)$ avec f la non linéarité choisie, W la matrice de paramètres de taille $N \times t$ et b un vecteur de biais de taille N. La couche de convolution équivalente est une couche à N filtres de convolution, chacun ayant un noyau de taille $n \times n$, chaque filtre de convolution étant suivi de la non linéarité f. Cette nouvelle couche équivalente de convolution peut donc être appliquée sur une image de plus grande taille que $n \times n$. Elle donne en sortie non plus un vecteur de N activations mais une carte spatiale d'activation pour laquelle on obtient en chaque position (i, j), le vecteur $y(i, j)$ de N activations prenant en compte une information locale des entrées sur un voisinage $n \times n$. En convertissant de cette façon un réseau de classification en son équivalent convolutionnel, il est possible alors de classer chacun des pixels de l'image traitée et obtenir ainsi une carte

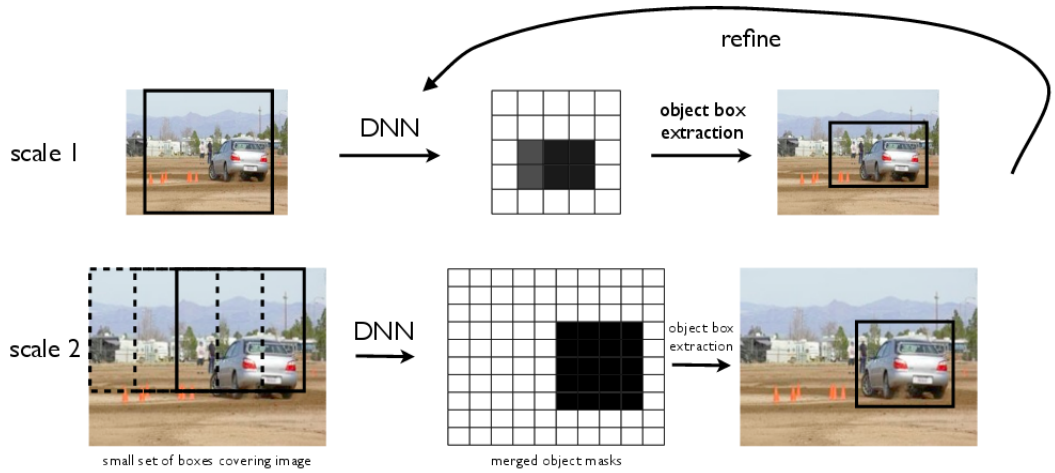


FIGURE 3.4. – Détection d'objets à l'aide de la méthode CNN : images extraites de [45].

de segmentation sémantique. Cette carte peut être de basse résolution en modifiant le pas d'échantillonnage spatial de ce nouveau réseau. L'intérêt de cette méthode, contrairement à la classification des différents patchs de l'image, est que l'image entière est soumise une seule fois au réseau de neurones qui va classifier chaque patch en profitant d'un partage des calculs pour les zones qui se chevauchent.

Les auteurs de l'article [44] se sont basés sur cette méthode et montrent qu'ils se placent à la première position en localisation et détection au moment de la parution de leur article sur la base de données ILSVRC.

Pour améliorer les performances de localisation, [45] propose une simple méthode de régression qui permet d'affiner la localisation d'un objet dans une image. En appliquant une première fois le CNN sur l'image d'entrée, une première boîte englobante permet de localiser un objet donné. La boîte englobante est affinée en répétant la même procédure sur l'image rognée, comme illustrée sur la Fig. 3.4 .

3.2.7.b. Utilisation des réseaux de classification

Les auteurs de [46], dans l'objectif de vérifier s'il était possible d'étendre les résultats de classification d'un réseau de neurones convolutionnels sur ImageNet [35] à un problème de détection d'objets sur la base de PASCAL VOC Challenge, propose une nouvelle approche nommée R-CNN (*Regions with CNN features*). Cette méthode comporte trois étapes, comme illustré sur la Fig.3.5 :

— extraction d'environ 2000 propositions de régions indépendantes, à l'aide d'une méthode dite Recherche Sélective [47]. Cette méthode, en envisageant des fenêtres de taille différente, cherche à mettre en évidence des zones constituées de pixels connexes présentant des caractéristiques (texture, couleur...) pouvant correspondre à un objet (c'est une forme de localisation d'objet) ;

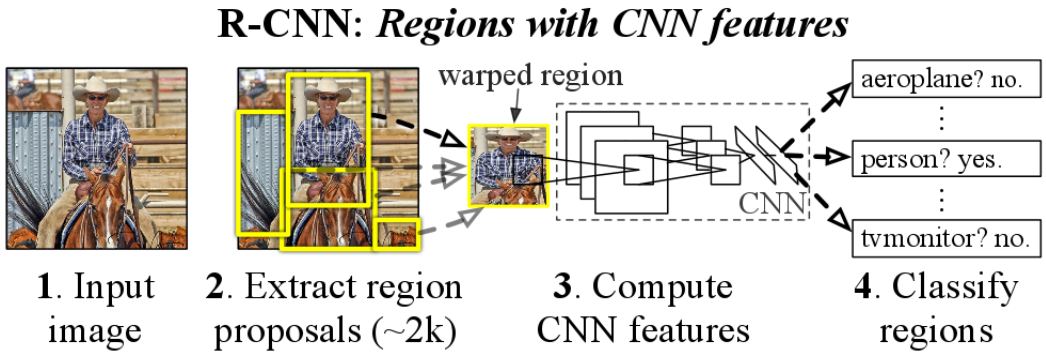


FIGURE 3.5. – Différentes étapes de la méthode R-CNN : images extraites de [46].

— chaque zone envisagée est ensuite redimensionnée dans une taille carrée standard et soumise à un réseau CNN (AlexNet modifié) afin de calculer les caractéristiques de chaque région. Il en résulte alors un vecteur de caractéristiques de taille fixe (4096) ;
— ensuite une classification de chaque zone est réalisée par un classifieur de type SVM, pour déterminer si la zone proposée contient un objet, et si oui lequel.

Les résultats obtenus sont meilleurs (mAP - mean Avereage Precision - de 43,5%) que ceux des autres méthodes basées sur les sacs de mots (par exemple [47] qui obtient une mAP de 35.1%).

Cependant, cette méthode, bien qu'elle atteigne de très bonnes performances en détection d'objets, présente quelques inconvénients comme le soulignent les auteurs de [48]. En particulier, la méthode R-CNN est particulièrement lente parce que chaque région proposée est soumise au réseau de neurones convolutionnel sans partage de calculs. De plus l'apprentissage d'un tel système est très coûteux en temps, car il opère de manière séquentielle. Dans [48], il est proposé une solution aux inconvénients précédents tout en améliorant les performances. Cette méthode, appelée « Fast R-CNN », extrait les caractéristiques de l'image d'entrée en la soumettant une seule fois au CNN. À partir du résultat obtenu, les caractéristiques de chacune des régions proposées sont extraites, en utilisant une méthode (couche du CNN) connue sous le nom de RoIPool (Region of Interest Pooling). Cette couche consiste à redimensionner les entrées qui ont une taille arbitraire (proposition de régions) en une sortie de taille fixe (en raison de la contrainte de taille dans les couches complètement connectées), comme on peut le voir sur la Fig.3.6. Donc il suffit d'un unique passage de l'image au travers du CNN, au lieu de 2000 passages pour l'algorithme R-CNN.

D'autres méthodes ont été proposées dans le but d'améliorer la rapidité de la détection en partageant au maximum les calculs, comme « R-FCN » [49] , « SSD » [50], Mask R-CNN [51] où encore celle nommée « Faster R-CNN », [52]. Cette dernière méthode se débarrasse de la première partie qui consiste à extraire des régions indépendantes, à l'aide

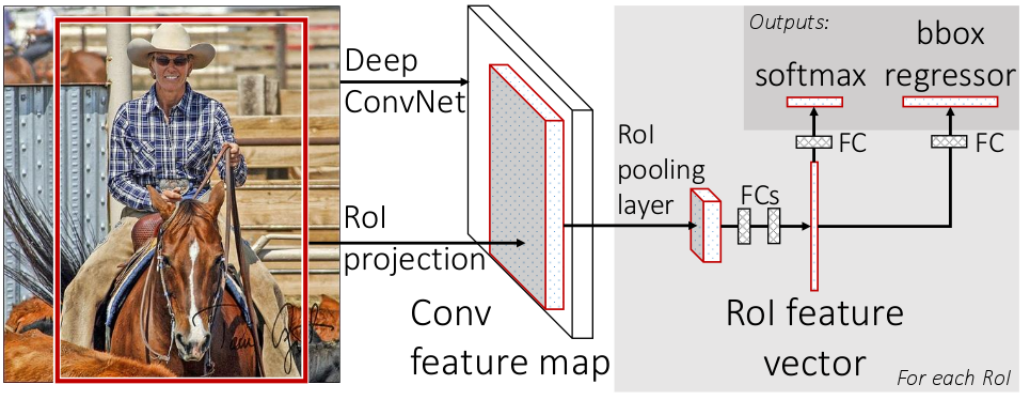


FIGURE 3.6. – Différentes étapes de la méthode FastR-CNN : images extraites de [48].

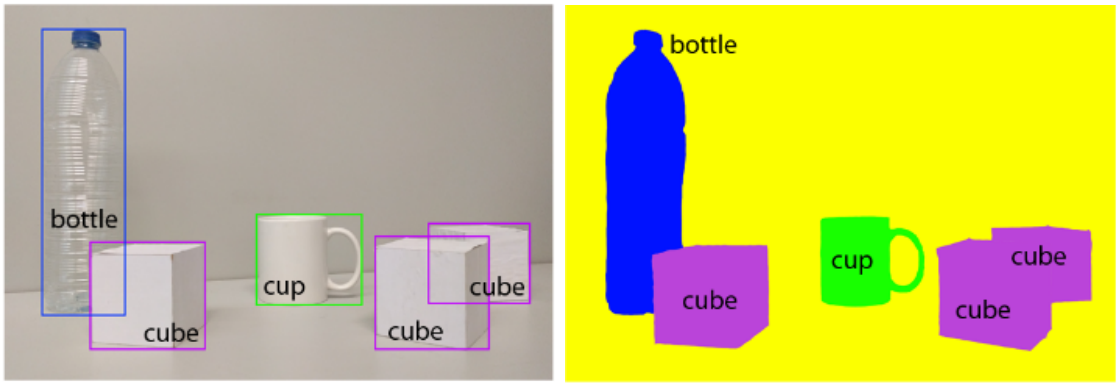
de la méthode dite Recherche Sélective afin de mettre en place un modèle complètement entraînable. Cette étape est remplacée par une couche nommée « RPN » (*Region Proposal Network*) qui a pour objectif de générer des régions susceptibles de contenir un objet.

En conclusion, il existe de nombreuses méthodes en matière de détection d'objets basées sur le Deep Learning, aussi bien dans des applications inédites que dans de nouvelles méthodes pour pousser les résultats de l'état de l'art. Ces différentes méthodes peuvent, dans notre cas, nous permettre de détecter différents objets, notamment le ticket de caisse dans l'image, mais aussi le logo et les zones de texte. Pour ces deux derniers (logo et zones de texte), nous allons développer dans les sections suivantes les différentes approches existantes dans ces domaines précis. Avant cela, nous allons aborder la segmentation sémantique qui permet une détection fine (au pixel près) de tous les objets d'une image.

3.2.8. Segmentation Sémantique

Introduction La segmentation sémantique consiste à comprendre une image au niveau du pixel, c'est-à-dire que l'objectif est d'attribuer à chaque pixel de l'image une classe d'objets. Au-delà de la détection et reconnaissance des différents objets dans l'image, la segmentation sémantique consiste à délimiter précisément les contours de chaque objet. Comme le montre la Fig.3.7, la segmentation sémantique est plus précise que la détection d'objets qui ne cherche à définir que la boîte englobante aussi précise soit-elle. Ce besoin d'analyse fine répond à un besoin de précision sur la partie avale décisionnelle de la chaîne de traitement. Cela peut être nécessaire dans certains domaines tels que la conduite autonome, les systèmes de réalité augmentée, etc.

Avant l'arrivée du Deep Learning, la segmentation sémantique était réalisée à l'aide d'approches telles que les « TextonForest » [53] ou les « Classifieurs basés sur les forêts aléatoires » [54]. Les « TextonForest » sont des forêts de décision aléatoires qui n'utilisent que de simples comparaisons de pixels sur des patchs locaux d'images, effectuant à la



(a) Détection d'objets. (b) Semantique Segmentation.

FIGURE 3.7. – Différence entre détection d'objets et segmentation sémantique

fois un regroupement hiérarchique implicite en textons sémantiques et une classification locale explicite de la catégorie de patch.

Les approches par Deep Learning, quant à elles, se basent quasiment exclusivement sur les réseaux de neurones convolutionnels (CNN). Contrairement aux réseaux CNN de classification, dans les réseaux réalisant une segmentation sémantique, le besoin de conserver une information spatiale précise limite fortement l'usage de couches de neurones complètement connectées. Néanmoins, les architectures proposées s'inspirent fortement des architectures de classification, sur leur partie convolutionnelle.
En 2014, apparaissent les premiers réseaux entièrement convolutionnels (FCN) [55] capables de segmenter efficacement des images complexes telles que celles du challenge ImageNet.

La principale difficulté de ces réseaux de segmentation est de répondre au besoin de représenter les données à différentes échelles et niveaux d'abstraction en sous-échantillonnant les images tout en garantissant une segmentation précise à l'échelle initiale. Les couches de max agrégation permettent ces changements d'échelle, mais elles s'accompagnent d'une perte d'information spatiale ce qui rend difficile la classification au niveau pixel à l'échelle originale.

Nous présentons ici deux classes d'architectures « codeur-décodeur » différentes qui ont été proposées dans la littérature pour s'attaquer à ce problème. Mais de manière générale, toutes les différentes approches prennent un réseau pour la classification et suppriment généralement ces couches entièrement connectées. Cette partie du nouveau réseau de segmentation reçoit souvent le nom d'encodeur et produit des représentations d'images à basse résolution ou des cartes de caractéristiques. Le problème réside dans l'apprentissage du décodage ou de la mise en correspondance de ces images à basse résolution avec des prédictions au pixel près pour la segmentation.

1 - Architectures avec couches de Max-Unpooling : La première est l'architecture utilisant les couches de max-unpooling. L'encodeur réduit progressivement la dimension spatiale avec des couches de max-agrégation et le décodeur récupère progressivement les détails de l'objet et la dimension spatiale. Il y a généralement des connexions de concaténations entre l'encodeur et le décodeur, pour une même résolution, afin d'aider le décodeur à mieux récupérer les détails de l'objet. Cela permet au décodeur de réapprendre des caractéristiques pertinentes qui sont perdues lorsqu'elles sont regroupées dans le codeur. U-Net est une architecture populaire de cette classe, comme illustrée sur la figure 3.12.

L'une des premières architectures CNN pour la segmentation sémantique, basée sur le principe d'encodeur-décodeur, et qui a obtenu des résultats intéressants est le réseau nommé SegNet [56]. L'architecture de la partie encodeuse a la même topologie que le réseau VGG-16 (avec 13 couches de convolutions). La nouveauté de SegNet réside dans la manière dont le décodeur sur-échantillonne ses cartes d'activation. Le décodeur utilise des indices des couches de max-agrégation calculés dans l'étape d'encodage à résolution correspondante pour effectuer un sur-échantillonnage non linéaire (≪ unpooling ≫), comme on peut le voir sur la Fig.3.8. Cela facilite le besoin d'apprendre à sur-échantillonner et par conséquent de diminuer le nombre de paramètres entraînables du réseau. Les cartes sur-échantillonnées sont ensuite convoluées avec un ensemble de filtres pouvant être entraînés pour produire des cartes de caractéristiques denses. Lorsque les cartes de caractéristiques ont été restaurées à la résolution d'origine, elles sont transmises au classificateur softmax pour produire la segmentation finale. Les résultats obtenus par SegNet étaient encourageants aussi bien en termes de précision que de temps de calcul.

2 - Architectures avec des convolutions dilatées : Les architectures de la seconde classe utilisent ce que l'on appelle des convolutions dilatées au lieu des couches de max-unpooling. Les convolutions dilatées permettent une augmentation du champ de vision sans diminution des dimensions spatiales.

Yu et al. dans [58] proposent d'utiliser la convolution dilatée. Dans [58] les deux dernières couches de max-agrégation du réseau de classification (ici, VGG-16) sont supprimées et les couches convolutives suivantes sont remplacées par des convolutions dilatées. Ils montrent, en évaluant leur architecture sur la base de données de VOC-2012, que l'utilisation de convolutions dilatées est adaptée à la segmentation sémantique. Ils montrent également que la suooression des composants des CNNs (couches de max-agrégation) pour la classification permet d'améliorer la précision des CNNs pour la segmentation sémantique.
D'autres travaux importants ont mis en évidence l'apport des couches de convolutions dilatées pour la segmentation sémantique, comme DeepLab [59] et ENet [60]. Tous

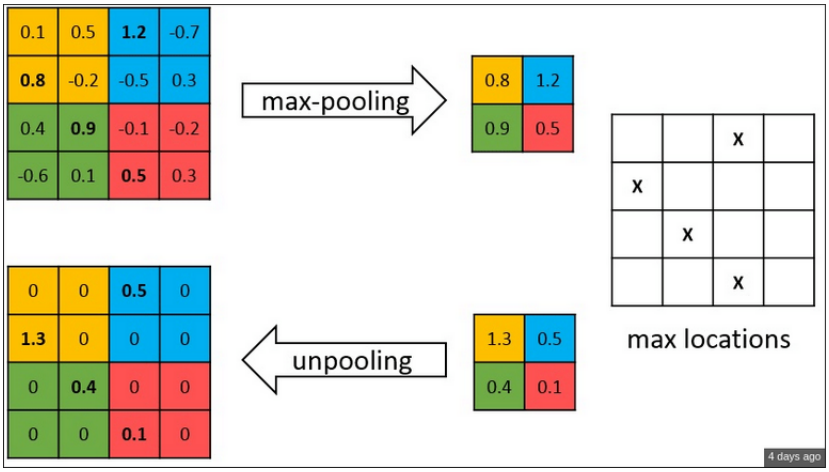


FIGURE 3.8. – Méthode de sur-échantillonnage (max-unpooling), image extraite de [57]. Pour chaque couche de max-agrégation, la position des maximums est sauvegardée. Après un processus réalisé par les couches neuronales intermédiaires, ces positions sont ré-utilisées dans les couches de max-unpooling.

utilisent des combinaisons de convolutions dilatées avec des taux de dilatation croissants pour avoir des champs de vision plus larges sans coût supplémentaire et sans trop sous-échantillonner les cartes de caractéristiques. Ces travaux montrent également une tendance commune : les convolutions dilatées sont étroitement liées à l'agrégation de contexte à plusieurs échelles dans les premières couches du réseau.

L'architecture ENet est composée de plusieurs étapes comme on peut les voir sur la Fig.3.9. Les trois premières étapes correspondent à la partie encodeuse, tandis que les deux dernières étapes appartiennent à la partie décodeur. L'apport de cette architecture est que pour une précision comparable aux autres méthodes de la littérature, ENet est beaucoup plus rapide (facteur 18) et requiert 80 fois moins de paramètres que les autres architectures existantes, car elle s'appuie sur des approches de type ResNet [38].

En 2016, Jegou et al. [61] ont étendu le réseau DenseNet [62] pour traiter le problème de la segmentation sémantique. Ils s'inspirent de DenseNet [62] pour la classification et portent ce réseau sur la tâche de segmentation en reprenant l'approche proposée dans la construction du réseau Unet [63] et en concaténant les cartes de caractéristiques de la partie encodage aux cartes sur-échantillonnées provenant du décodeur à chaque étape. L'architecture, par ces connexions de concaténation, permet au décodeur à chaque étape de réapprendre des caractéristiques pertinentes qui sont perdues lorsqu'elles sont agrégées dans le codeur, comme illustrée sur la Fig. 3.12.

Name	Type	Output size
initial		$16 \times 256 \times 256$
bottleneck1.0	downsampling	$64 \times 128 \times 128$
$4\times$ bottleneck1.x		$64 \times 128 \times 128$
bottleneck2.0	downsampling	$128 \times 64 \times 64$
bottleneck2.1		$128 \times 64 \times 64$
bottleneck2.2	dilated 2	$128 \times 64 \times 64$
bottleneck2.3	asymmetric 5	$128 \times 64 \times 64$
bottleneck2.4	dilated 4	$128 \times 64 \times 64$
bottleneck2.5		$128 \times 64 \times 64$
bottleneck2.6	dilated 8	$128 \times 64 \times 64$
bottleneck2.7	asymmetric 5	$128 \times 64 \times 64$
bottleneck2.8	dilated 16	$128 \times 64 \times 64$
Repeat section 2, without bottleneck2.0		
bottleneck4.0	upsampling	$64 \times 128 \times 128$
bottleneck4.1		$64 \times 128 \times 128$
bottleneck4.2		$64 \times 128 \times 128$
bottleneck5.0	upsampling	$16 \times 256 \times 256$
bottleneck5.1		$16 \times 256 \times 256$
fullconv		$C \times 512 \times 512$

FIGURE 3.9. – Architecture du réseau ENet, images extraites de [60].

Le réseau FC-DenseNet est construit à partir de blocs denses et d'opérations de « max-unpooling », dont la composition est décrite dans le schéma 3.11. Chaque bloc dense est une concaténation itérative des cartes de caractéristiques des couches précédentes comme illustrée sur la figure 3.10.

Dans l'article [61], le réseau évalué est constitué de cinq blocs denses de taille croissante pour la partie sous-échantillonnage et l'extraction des caractéristiques comme illustré dans le tableau 3.1. Naturellement la partie de sur-échantillonnage permettant de retrouver la résolution de l'image de départ est également composée de cinq blocs denses.

L'architecture finale est très profonde, car elle contient 103 couches, mais 10 fois moins de paramètres que les autres modèles de la littérature. La précision est bien meilleure puisque sur la base de données CamVid, ce réseau permet d'obtenir une IoU de 66.9% contre 46,4% avec SegNet.

3.2.9. Apprentissage multi-tâche

L'apprentissage multi-tâche consiste à conjecturer de manière simultanée plusieurs modèles de prédictions. L'intuition est qu'une prédiction simultanée permet de faire mieux qu'une prédiction indépendante si les tâches partagent des similarités.

Layer
Batch Normalization
ReLU
3×3 Convolution
Dropout $p = 0.2$

Transition Down (TD)
Batch Normalization
ReLU
1×1 Convolution
Dropout $p = 0.2$
2×2 Max Pooling

Transition Up (TU)
3×3 Transposed Convolution $stride = 2$

FIGURE 3.10. – Les blocs de construction de FC-DenseNet entièrement convolutionnels. De gauche à droite : « Layers » : couche utilisée dans le modèle, « Transition Down » transition de sous-échantillonnage et « Transition Down » transition de sur-échantillonnage, image extraite de [61].

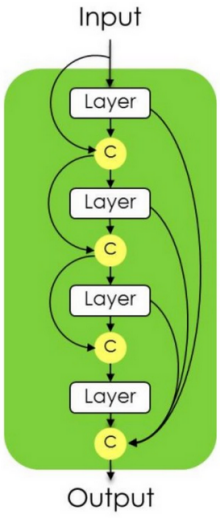
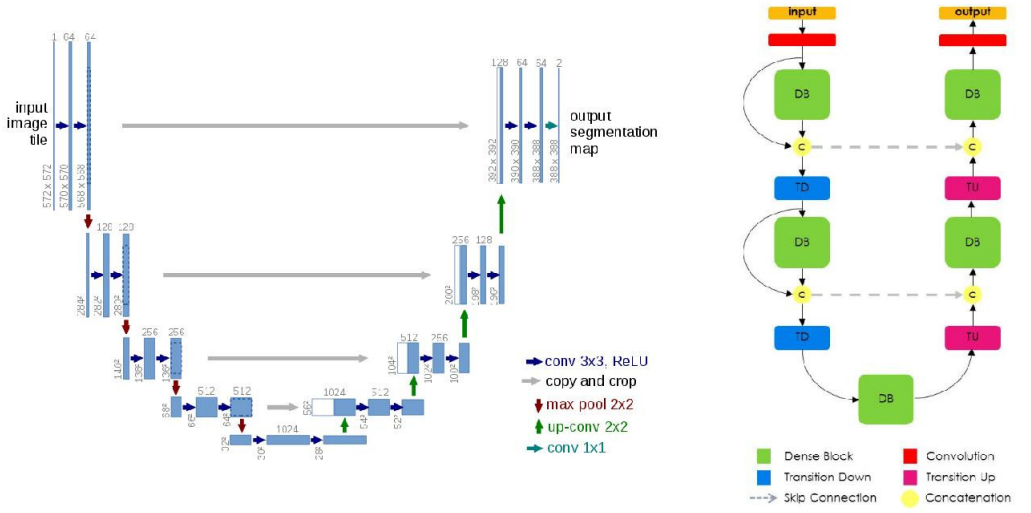


FIGURE 3.11. – Diagramme d'un bloc dense de 4 couches du réseau FC-Densenet, image extraite de [61].

Architecture
Image en entrée, $m = 3$
Convolution 3x3, $m = 48$
DB (4 couches) + TD, $m = 112$
DB (5 couches) + TD, $m = 192$
DB (7 couches) + TD, $m = 304$
DB (10 couches) + TD, $m = 464$
DB (12 couches) + TD, $m = 656$
DB (15 couches), $m = 48$
TU + DB (12 couches), $m = 48$
TU + DB (10 couches), $m = 816$
TU + DB (7 couches), $m = 578$
TU + DB (5 couches), $m = 384$
TU + DB (4 couches), $m = 256$
Convolution 1x1, $m = c$
Softmax

TABLE 3.1. – Architecture FC-DenseNet présentée dans le papier [61] de 103 couches de convolutions. « DB » : Blocs Denses, « TD » : Transitions de sous-échantillonnage, « TU » : Transitions de sur-échantillonnage , « m » : correspond au nombre total de cartes de caractéristiques (*features map*) et « c » correspond au nombre de classes.



Architecture U-Net **Architecture FC-DenseNet**

FIGURE 3.12. – Architecture de U-Net à gauche, image extraite de [63] en 2015 et architecture de FC-DenseNet à droite image extraite de [61] en 2016.

Dans plusieurs domaines, l'apprentissage multi-tâche a été un succès, par exemple, dans le traitement naturel du langage [64], en analyse d'image [65] ou encore pour la reconnaissance de la parole [66]. L'une des premières motivations à mettre en place de l'apprentissage en multi-tâche est inspirée par la méthode d'apprentissage des êtres humains. En effet, dans l'apprentissage de tâches, l'homme utilise les connaissances acquises dans divers domaines pour apprendre de manière plus efficace une nouvelle tâche. Du point de vue de l'apprentissage automatique, le multi-tâche est une forme de transfert inductif. Les tâches intermédiaires permettent d'amener le modèle à préférer certains exemples afin de dégager des règles les plus générales possible pour reconnaître une catégorie d'objets. Or la capacité d'un réseau à généraliser est l'une des qualités recherchées en classification. L'article [67] donne un aperçu de l'apprentissage multi-tâche dans les réseaux de neurones profonds.

L'article [67] présente deux approches différentes pour faire de l'apprentissage multi-tâche avec les réseaux de neurones profonds. La méthode la plus répandue avec « partage de paramètres en dur » et une autre méthode où le « partage de paramètres est léger » dans le sens ou les paramètres d'un réseau d'une tâche précise vont influencer les paramètres du réseau d'une autre tâche. Sur la figure 3.13, est illustrée la différence entre les deux approches. Ces architectures multi-tâches ont l'avantage de réduire considérablement le sur-apprentissage. Dans la suite de l'article, les travaux récents basés sur l'apprentissage multi-tâche en Deep Learning sont présentés, tels que [68, 69, 70]. Par exemple [68] propose une architecture multi-tâche avec, entre chaque couche connectée, des matrices afin d'apprendre les relations entre chaque tâche. Alors que [69] prend en considération l'incertitude de chacune des tâches dans le calcul du « loss ».

3.3. La détection de logos

3.3.1. Introduction

Dans la littérature, la détection et la reconnaissance de logos dans des images ont largement été étudiées. Cette problématique, qui peut être considérée comme un cas particulier de détection d'objets, possède cependant ses propres spécificités, et ses propres challenges. La détection de logos couvre un large éventail d'applications. Par exemple : identification de marques, mesure de la visibilité d'une marque dans une rencontre sportive, recherche de logos de véhicules pour le transport intelligent, recherche de logos dans les images postées sur les réseaux sociaux détermination (élément contribuant à la définition du profil-utilisateur)...

Dans le cas particulier de l'analyse d'image de documents, la détection de logos présente un grand intérêt, car elle permet d'identifier la provenance du document et d'utiliser ainsi des informations a priori sur la structure de ce document, facilitant ainsi

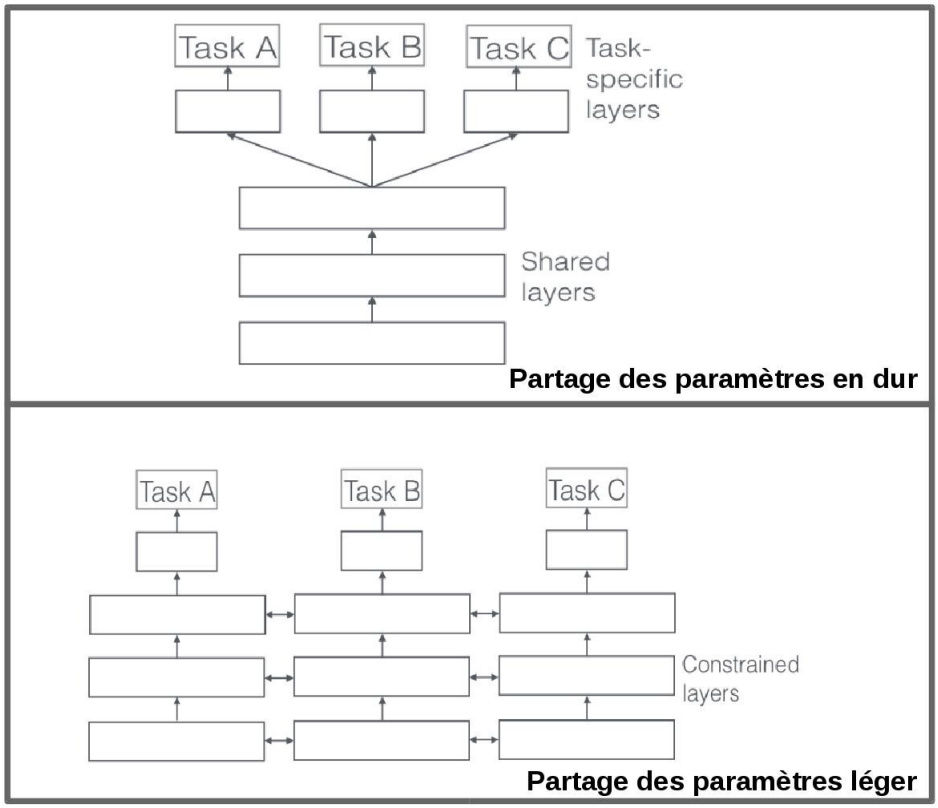


FIGURE 3.13. – Les deux différentes approches permettant de réaliser du multi-tâche avec des réseaux de neurones profonds : « partage de paramètres en dur » et « partage de paramètres léger ». Ces illustrations sont extraites de l'article [67] en 2017.

son analyse. Ceci est particulièrement vrai dans le cas des tickets de caisse où la connaissance de l'enseigne peut permettre d'exploiter la connaissance de la fonte de caractères utilisée, la position des différents blocs de texte dans le ticket..., autant d'informations pouvant simplifier l'analyse.

Les logos sont généralement constitués de texte et de symboles graphiques. Par rapport à la détection d'objets quelconques, la recherche de logos peut paraître plus simple, car, de par leur nature, les logos ne doivent pas présenter une grande variabilité. Cependant, un certain nombre de phénomènes (petite taille, mauvais éclairage, rotation, occlusion partielle...) compliquent souvent leur détection.

On trouve une illustration de ce problème dans la tâche INstance Search (INS) du challenge TRECVid [1] qui propose de détecter des cibles, dont des logos, dans des scènes complexes sachant que ces cibles ne sont que des objets secondaires dans les scènes analysées. Les niveaux de performances dans ce type de tâches, pour lesquelles on ne dispose que de peu d'exemples de cibles, restent bas (de l'ordre de 30% de précision moyenne lors des dernières éditions [71]).

Dans la littérature, plusieurs approches pour la détection de logos ont été proposées. Classiquement, la détection de logos a été abordée en utilisant des descripteurs SIFT et des classifieurs du type SVM. Puis l'introduction des sacs de mots a permis la création de vocabulaires à partir des descripteurs SIFT et d'améliorer les performances de détection des logos. Dans ces travaux, seuls de petits ensembles de données de logos sont évalués, avec un nombre limité d'images de logos et de classes modélisées. Plus récemment, quelques travaux ont exploré l'utilisation des réseaux de neurones et plus précisément du Deep Learning. Nous allons, dans cette section, présenter les différentes approches existant dans la littérature, en commençant tout d'abord par présenter rapidement les bases de logos utilisées.

3.3.2. Bases de logos

Bien que les travaux concernant la reconnaissance de logos existent depuis plusieurs années, la plupart d'entre eux utilisent de petits ensembles de données, les bases de données plus grandes n'étant pas publiques. Parmi les différentes bases d'images de logos annotées existantes, l'une des plus grandes, et donc celle qui est la plus utilisée, est FlickrLogo-32.

3.3.2.a. Ensemble de données FlickrLogos

L'ensemble de données FlickrLogos se compose d'images acquises de façon non standardisée recueillies à partir de Flickr représentant des logos de marques dans diverses circonstances. FlickrLogos-32 a été conçu pour la détection et la reconnaissance de logos multiclasses enfouis dans des images de scènes naturelles.

Cette base est constituée de 32 classes de logos pour un total de 8240 images, dont 5644 contenants des logos. Certaines images peuvent contenir plusieurs instances d'un même

1. https://trecvid.nist.gov/

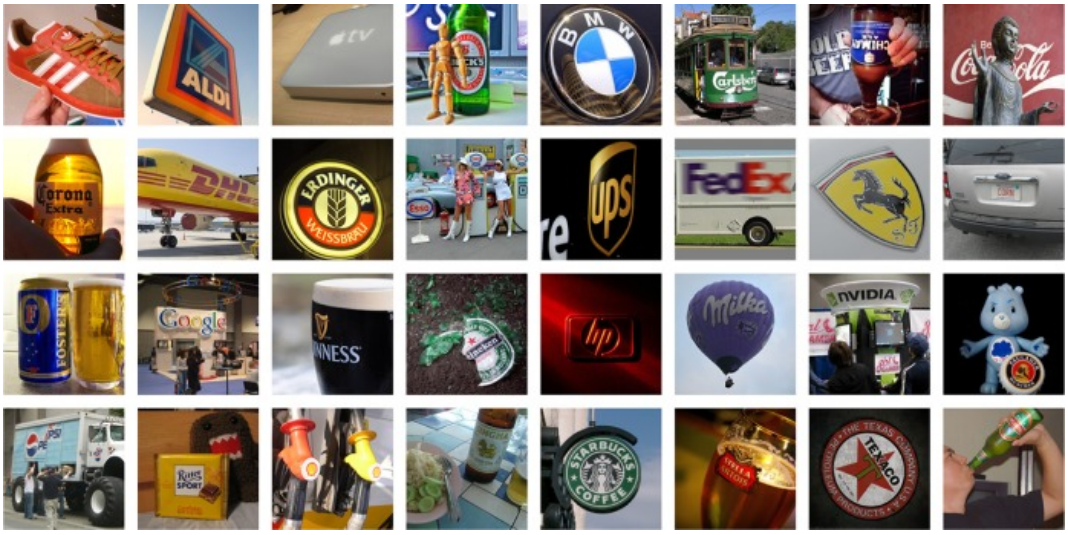


FIGURE 3.14. – Exemples d'images de logos de la base de données Flickr-32.

logo. FlickrLogos-32 est divisée en trois sous-ensembles, un ensemble pour l'entraînement avec 10 exemples de logos par classe et deux sous-ensembles pour la validation et la phase de test contenant chacun 30 exemples de logos par classe. Comme il est important d'évaluer également la sensibilité sur des images ne contenant aucun logo, ces deux derniers sous-ensembles contiennent également 3000 images sans logo.

Depuis le mois de mai 2017, une nouvelle version de cette base est disponible il s'agit de FlickrLogos-47. Comme son nom l'indique, cette base contient 47 classes de logos. Elle a été construite à partir des mêmes images que la base FlickrLogos-32, mais les annotations ont été reprises de façon à avoir une annotation plus précise (en effet, dans la base FlickrLogos-32 seules les instances de logo les plus importantes étaient étiquetées). Cependant, malgré l'ajout de 15 nouvelles classes cette base reste encore trop petite pour pouvoir explorer les approches de type Deep Learning qui demandent une base d'apprentissage de très grande taille.

3.3.2.b. Logo-Net

La base LOGO-Net Deep :LogoNet a été proposée en novembre 2015. Cette base contient deux sous-ensembles. Un premier sous-ensemble contenant 18 classes de logos et 16 043 objets logos. Le deuxième sous-ensemble, beaucoup plus grand (73 414 images), contient 160 classes de logos et 130 608 objets logos. Cette base de données a été manuellement annotée avec des fenêtres englobantes encadrant chaque logo permettant ainsi d'optimiser des détecteurs sur une tâche de localisation. Globalement, cette base permet ainsi de comparer de façon fiable les performances de différentes méthodes sur des tâches de détection, classification et localisation.

FIGURE 3.15. – Exemples d'images de logos de la base de données LOGO-Net.

3.3.2.c. Propositions de nouvelles bases de données pour le Deep Learning

Les travaux les plus récents se concentrent plus sur la recherche d'amélioration et d'enrichissement des bases d'entraînement, plutôt que sur les méthodes de détection elles-mêmes. Su et al. présentent la base « TopLogo-10 » dans l'article [72] daté de janvier 2017. Cette base, contenant 10 classes de logos, a été construite à l'aide d'un nouvel algorithme de génération d'images. L'apport de cette approche est essentiellement dans l'augmentation automatique des données d'apprentissage. Les mêmes auteurs proposent également une autre base de données, beaucoup plus grande [73], contenant 194 classes de logos et près de 2 millions d'images. Cet ensemble de données a été construit automatiquement en échantillonnant les données du flux Twitter. Cependant, ces images sont bruitées et les annotations sont faibles. Les auteurs de [74] ont également proposé une extension de la base FlickrLogo-32, qu'ils ont nommée « Logos-32plus », contenant environ 400 exemples par classe au lieu des 70 de FlickrLogo-32. Ce sont les seuls, à notre connaissance, qui ont entraîné un CNN entièrement (sans utilisation de poids pré-entraînés) pour faire de la détection de logos. Compte tenu de la petite taille de leur base d'entraînement, ils ont défini une architecture de réseau contenant trois couches de convolutions et deux couches complètement connectées, donc un CNN peu profond quand on le compare aux CNNs les plus performants (cf 3.2.5). Ils ont conçu un pipeline de reconnaissance complet comprenant une proposition de régions candidates qui sont soumises au CNN. Les expériences menées montrent que les résultats sont meilleurs avec la nouvelle base de données.

3.3.3. Application des descripteurs SIFT à la détection de logos

De nombreux chercheurs se sont intéressés à l'utilisation de descripteurs SIFT pour détecter et reconnaître des objets et notamment des logos dans des images de scènes naturelles. La particularité de ces descripteurs est leur robustesse aux changements d'intensité, aux variations d'échelle et aux rotations, ce qui répond parfaitement aux difficultés que présente la détection de logos dans une image. Bagdanov et al. [75] ont considéré

un logo comme un ensemble de descripteurs SIFT pour détecter et reconnaître les logos dans les vidéos de sport. L'approche est basée sur la mise en correspondance entre un ensemble de descripteurs SIFT pour chaque logo de leur base de connaissance et l'ensemble des caractéristiques SIFT détectées dans chaque image de la vidéo. Cependant, leur approche ne s'adapte pas bien à une base de données de très grande taille.

3.3.4. Les sacs-de-mots visuels pour la détection de logos

Plusieurs travaux, postérieurs à [75], se sont basés sur l'intégration d'approches sacs de mots à des descripteurs SIFT [76, 77, 76, 78, 79], pour faire à la fois de la détection et de la classification de logos. L'approche par sacs de mots permet de quantifier dans un vocabulaire les caractéristiques locales décrites par les descripteurs SIFT. Romberg et al. [77] proposent une méthode d'indexation de la disposition spatiale des caractéristiques locales et de la composition des structures spatiales de base, telles que les bords et les triangles détectés dans les images du logo. Cette méthode permet d'obtenir une bonne précision, supérieure à 98% sur la base de test de FlickrLogos-32 mais le rappel reste faible (61 %).

Boia et al. [78, 79] proposent une approche pour de la détection de logos utilisant des graphiques homographiques de classes. Ils construisent des modèles de classe en faisant correspondre des caractéristiques locales, toujours extraites à l'aide des descripteurs SIFT, entre des exemples de la même classe pour créer des homographies. De cette façon chaque classe est représentée par un ensemble de points d'intérêts et un modèle de caractéristiques, ce qui permet d'obtenir une grande précision du système de reconnaissance de logos proposé. Ils ont évalué leur méthode sur la base de données FlickrLogos-32 et les résultats obtenus dépassent ceux de la méthode proposée par Romberg et al. [77].

3.3.5. Deep Learning pour la détection de logos

Le succès du Deep Learning dans de nombreux domaines de la vision par ordinateur a incité à l'exploration de cette piste pour la détection de logos. Cependant, l'absence de grandes bases d'apprentissage dans ce domaine, nécessaires à l'entraînement des réseaux profonds, a freiné le développement de telles approches. Néanmoins, pour contourner ce problème, plusieurs solutions ont été proposées.

CNN comme extracteur de caractéristiques Une des solutions qui a été expérimentée par [80, 81] est l'utilisation d'un réseau de neurones convolutionnels comme un extracteur de caractéristiques. Les auteurs de [80] proposent de ré-utiliser un CNN pré-entraîné, de 16 couches, sur ImageNet (VGG-16 [36]). Ils considèrent la sortie de la première couche complètement connectée, ce qui permet d'extraire un vecteur de caractéristiques de dimensions 4096. Une fois ce vecteur normalisé il est utilisé pour entraîner des SVMs linéaires (ils utilisent un SVM pour chaque classe de logo). Afin d'obtenir de bonnes performances sur ce système, les auteurs ont dû générer des données synthétiques en

l'absence de base d'apprentissage assez grande (transformations perspectives, modifications de la couleur, de l'échelle et de l'arrière-plan).

En entraînant les classifieurs SVMs avec cette nouvelle base de données contenant des images générées synthétiquement, ils montrent que les performances sont meilleures. Mais ils considèrent qu'il est possible d'améliorer encore ces performances en utilisant ce processus d'augmentation de données pour entraîner directement un CNN.
De la même manière, Bianco et al. [81] ont expérimenté une méthode très proche de celle-ci (à savoir l'utilisation de CNNs et de données générées synthétiquement) en s'inspirant des travaux de Girshick [48]. En prenant en entrée une image, ils extraient des régions susceptibles de contenir un objet. Ces régions, appelées « propositions d'objets », sont redimensionnées en une taille fixe avant d'être soumises à un CNN pré-entraîné comme un extracteur de caractéristique. Enfin, un SVM linéaire permet la reconnaissance et la classification.

Transfert d'apprentissage Iandola et al. [82] proposent différentes architectures de CNN, en considérant trois variantes du problème de reconnaissance de logos : i) la classification, ii) la détection sans localisation et iii) la détection avec localisation. Pour la classification (i), les architectures testées sont au nombre de trois et s'inspirent de l'architecture de GoogLeNet [37] :

- GoogLeNet-GP (GoogLeNet with Global Pooling), pour cela ils ajoutent simplement une couche de « average-pooling » avant les couches complètement connectées, cela permettant au réseau de prendre en entrée une image de taille quelconque aussi bien pendant la phase d'entraînement que pendant la phase de test ;
- GoogLeNet-FullyClassify : comme on l'a vu au paragraphe 3.2.5, GoogLeNet a trois classifieurs softmax situés après Inception3, Inception6 et Inception9. Iandola et al. proposent de pousser cette idée à l'extrême et de placer un classifieur Softmax après chaque couche de type Inception. Pendant la phase de test, seule la dernière couche de Softmax est utilisée ;
- Full-Inception : dans ce réseau, les auteurs remplacent la première couche en un module d'Inception. Ce réseau vise à améliorer la classification des logos à basse résolution pour de grandes images en ayant une gamme de tailles de filtre dès la première couche qui agit sur les pixels d'entrée.

Pour entraîner tous ces réseaux, les auteurs utilisent la méthode du transfert d'apprentissage et ré-affectent les poids du réseau GoogLeNet entraîné sur ImageNet. Seules les nouvelles couches sont initialisées par des poids aléatoires et entraînées sur la base d'entraînement de FlickrLogos-32. Concernant la classification de logos (i), le réseau GoogLeNet-GP a permis d'atteindre des performances (précision de 89,6%) qui surpassent celles de l'état de l'art du moment. Pour la détection de logos sans ou avec localisation (ii et iii) les auteurs utilisent un réseau Fast R-CNN pour la détection ce qui permet (comme nous l'avons vu dans le paragraphe 3.2.7) de faire de la localisation d'objets avant la classification. Pour la classification, ils utilisent encore une fois du transfert d'apprentissage sur le réseau AlexNet [31] ou VGG16, sur la base d'entraînement de

FlickrLogos-32. Ils obtiennent des précisions de 73,3% pour la recherche sans localisation et de 74,4% pour la recherche avec localisation.

Dans un autre article, Oliveira et al [83] s'inspirent également de la méthode de Girshick [48], tout en tirant profit du transfert d'apprentissage. La méthode d'extraction de régions dites de propositions d'objets est identique à celle utilisée dans [48]. Deux CNNs pré-entrainés ont été testés en entraînant les dernières couches sur la base de FlickrLogos-32 (AlexNet [31] et VGG_CNN_M_1024 [84]). Ces deux réseaux permettent d'obtenir de bonnes performances. Cette méthode peut être améliorée en diminuant le nombre de faux positifs avec une méthode d'extraction de régions adaptée aux logos.

Face à ce problème d'absence de base de données, les auteurs de [85] introduisent une base d'images de logos à grande échelle destinée à faciliter l'implémentation des nouvelles méthodes pour la détection de logos à partir d'images du monde réel (base nommée « LOGO-Net » que nous avons déjà présentée dans le paragraphe 3.3.2.b). Hoi et al. ont expérimenté les nouvelles méthodes de détection d'objets basées sur les réseaux convolutionnels RCNN et FRCNN (vu au paragraphe 3.2.7). Malgré l'augmentation de la base de données, les réseaux utilisés par Hoi et al. (AlexNet et VGG-16) sont trop profonds pour être entraînés à partir de « LOGO-Net » seulement. Ils optent donc pour le transfert d'apprentissage. Les résultats sont satisfaisants puisqu'ils atteignent avec le réseau AlexNet une précision de 95%, ce qui est bien meilleur que la performance atteinte par Iandola et al. [82], en entraînant ce même réseau avec la base de FlickrLogos-32 (pour rappel Iandola et al. ont obtenu une précision de 89,6%).

3.3.6. Synthèse

Comme nous avons pu le voir, la détection de logos est un domaine très actif dans la littérature depuis déjà quelques années. Plusieurs approches basées sur différentes techniques ont été proposées, atteignant des performances encourageantes. Dans notre cas, les logos des tickets de caisses sont plus simples à détecter que ceux de la littérature (cf la base FlickrLogo-32) dans le sens où les variations intra-classe sont très faibles et le contexte direct est sensiblement le même (fond clair du ticket de caisse). Parmi les différentes approches, nous avons très vite abandonné les méthodes basées sur les descripteurs SIFT, car comme nous l'avons mentionné, ces derniers sont perturbés par la présence de patterns répétitifs, comme les caractères. Or les logos des enseignes sont essentiellement constitués de caractères, comme illustrée sur la fig. 3.16. Nous nous sommes donc intéressée aux méthodes basées sur le Deep Learning.

FIGURE 3.16. – Exemples d'images de logos de tickets de caisse.

3.4. Détection de zones de texte et et Reconnaissance de caractères

3.4.1. Introduction

La lecture des textes contenus dans une image (publicités, noms de rues, de magasins, panneaux routiers...) peut-être une source d'information très utile pour la compréhension du contenu de cette image. Dans ce cadre, la détection et la reconnaissance de texte dans des images sont des sujets de recherche importants et actifs dans la vision par ordinateur. Dans la littérature, on distingue deux situations : la détection de texte dans des scènes naturelles et la détection de texte dans des images de document. Dans les scènes naturelles, le texte est souvent décrit comme « enfoui » , car il peut ne représenter qu'une petite partie de l'image au milieu d'éléments très divers et apparaître sous des aspects (fonte, couleur, orientation...) très différents. À l'inverse, une image de document contient généralement essentiellement du texte dans un format plus standardisé. Dans notre cas, on devrait naturellement s'intéresser à la situation particulière de détection et reconnaissance de caractères dans une image de document. La détection de zones de texte dans une image de document a largement été étudiée dans la littérature et les techniques développées (fenêtre coulissante, projections), comme nous l'avons vu dans le chapitre 2, permettent d'obtenir de bons résultats. Les images de documents scannés sont très propres et sachant qu'un document contient une structure bien définie (paragraphes et colonnes), il est plus facile de trouver les zones de texte. Aujourd'hui,

FIGURE 3.17. – Acquisition d'images de ticket de caisse prise dans des conditions rendant la localisation de texte difficile.

plusieurs systèmes libres ou payants sont disponibles (Tesseract [19], ABBYY FineReader [20], Google Vision [21] par exemple). Cependant, ces méthodes n'ont généralement été développées et testées que sur des images de documents scannés où le document est plat, bien droit et sur fond uni, conditions qui sont difficilement réunissables dans notre cas, comme illustré sur la Fig. 3.17. C'est pourquoi nous avons également étudié les approches proposées dans la littérature concernant la détection de zone de texte dans des images naturelles.

La lecture de caractères dans des images est classiquement divisée en deux étapes : une première étape de détection des zones de texte (caractères, mots, lignes ou groupes de lignes), et une seconde étape de reconnaissance des caractères et/ou mots contenus dans les zones de texte détectées.

Dans la littérature, certains auteurs se sont focalisés seulement sur la détection de zones de texte [86, 87, 88], ou alors sur la reconnaissance du texte [89, 90], et d'autres approches proposent de combiner les deux [91, 92, 93, 94, 25].

Nous allons dans cette partie, présenter les différentes méthodes concernant la détection de zones de textes, puis nous présenterons les approches existantes concernant la reconnaissance de caractères. Enfin, nous nous intéresserons aux méthodes qui ont fait le choix de gérer conjointement les deux objectifs.

Avant d'aborder cet état de l'art, précisons l'existence de deux bases dédiées à la détection et à la reconnaissance de caractères. La première concerne les bases de données des challenges liés aux conférences ICDAR (International Conference on Document Analysis and Recognition). Ces challenges, qui se poursuivent annuellement depuis 2003, proposent des données pour couvrir un large éventail de situations du monde réel contenant du texte. La base IDCRA2003 contient par exemple une dizaine de milliers d'images

avec des annotations sous forme de boîtes englobantes.

L'ensemble de données « Street View Text » (SVT) contient des images, téléchargées à partir de « Google Street View », de scènes en bordure de route. Il comporte des annotations sous forme de boîtes englobantes de mots. Le texte étiqueté peut être très difficile à lire et à détecter avec une grande variété de polices, d'orientations et de conditions d'éclairage.

3.4.2. Détection de zones de textes

Les méthodes peuvent être catégorisées de la façon suivante :

3.4.2.a. Méthodes basées sur la détection de régions

Une des approches qui est largement utilisée est basée sur la détection de régions. Ces méthodes s'appuient sur les propriétés de couleur ou sur la variation des niveaux de gris dans les zones contenant du texte afin de marquer des différences par rapport aux propriétés correspondantes à l'arrière-plan. Ces méthodes sont fondées sur l'hypothèse qu'il y a très peu de variation de couleur dans le texte et cette couleur est suffisamment distincte du voisinage immédiat du texte.

Pour permettre d'évaluer les différences entre le texte et le fond, plusieurs méthodes sont proposées :

Détection de régions : Composante connexe, ou approches ascendantes Les méthodes basées sur les composantes connectées (*Connected Component*) segmentent les pixels de l'image en caractères, puis les regroupent en mot, en ligne, etc. C'est une méthode dite ascendante, car elle cherche à regrouper les petits éléments (caractères) en composants de plus en plus grands (mots, lignes, paragraphes) jusqu'à ce que toutes les zones de textes de l'image soient identifiées dans l'image. Généralement, une analyse géométrique est nécessaire afin de fusionner les différents composants et marquer les limites des zones de textes.

Par exemple, partant du principe que le cerveau humain est capable de reconnaître une zone de texte même s'il n'est pas capable de déchiffrer le message, Gomez et al. proposent dans [95] une méthode où les régions extraites sont regroupées de manière ascendante guidée par des preuves de similarité comme la couleur, la taille etc. dans le but d'obtenir des zones significatives qui pourraient être du texte. Cette méthode se décompose en trois grandes étapes :

1. décomposition en régions en utilisant la technique MSER (Maximally Stable Extremal Regions) ;
2. Extraction d'hypothèses de regroupements possibles pour obtenir des preuves ;
3. formation d'une ligne de texte et validation.

Neumann et Matas [96, 97] utilisent également la technique des MSER pour regrouper les caractères détectés en mots ou lignes de texte. Pour ce faire, l'algorithme proposé utilise plusieurs projections de l'image initiale. Le meilleur compromis entre rapidité d'exécution et efficacité est l'utilisation des projections H (hue/teinte), S (saturation/saturation), I (intensity/intensité) et de leur négatif et d'un gradient de l'intensité.

Les différentes projections obtenues sont ensuite soumises à deux classifieurs. Le premier est un classifieur AdaBoost qui permet de détecter les régions contenant un caractère, en mesurant la probabilité de chaque région d'être un caractère à partir de caractéristiques très peu coûteuses en calcul, comme le périmètre, la surface, la boîte englobante ou encore le nombre d'Euler. Les régions sélectionnées par ce premier classifieur sont soumises à un second classifieur, un SVM à noyau RBF. Cette fois-ci, plus de caractéristiques sont prises en compte, ce qui est plus coûteux en calcul, pour augmenter la précision et donc diminuer le nombre de faux positifs. Une fois que les régions extrêmes correspondant à un caractère sont toutes détectées, ils utilisent un algorithme basé sur la technique MSER (Maximally Stable Extremal Regions) pour regrouper ces caractères en zone de texte.

Détection de régions : Fenêtres glissantes Une autre approche basée sur la détection de régions consiste à utiliser des fenêtres glissantes (méthodes rapidement évoquées au chapitre 2).

Par exemple, Wang et al [98] effectue la détection de caractères par classification de fenêtres glissantes multi-échelle. Un classifieur est appliqué sur une fenêtre glissante pour trouver les caractères dans une image. Les auteurs de l'article [99] proposent également une méthode permettant de détecter les caractères dans une image par une méthode basée sur les fenêtres glissantes. Dans cette méthode, une série de prétraitements leur permet de mettre en évidence les contours des objets dans l'image et d'estimer la taille des différentes régions ainsi localisées. Pour chaque taille de région détectée, une fenêtre glissante parcourt l'image afin d'analyser chaque région et localiser celle qui correspond à des zones de textes. Les résultats des expérimentations de ces méthodes montrent des performances intéressantes.

Ces méthodes, bien qu'ayant fait leurs preuves dans de nombreux cas d'études, restent néanmoins très coûteuses en termes de calcul. De plus, elles ne garantissent pas la détection de tous les blocs de texte. En particulier, l'efficacité est fortement réduite lorsque l'arrière-plan est complexe.

3.4.2.b. Projections, ou Approches descendantes

D'autres méthodes suivent des approches descendantes en divisant l'image en parties de plus en plus fines. On note que ces méthodes sont surtout utilisées dans les images de documents [100, 17, 101]. L'algorithme proposé par [17] recherche des espaces blancs verticaux et horizontaux pour diviser successivement les pages en paragraphes, en lignes et en mots. C'est la méthode que nous avons présentée dans le chapitre 2. La localisation des séparations blanches est généralement effectuée en analysant le profil de la courbe

d'intensité de pixels blancs/noirs dans une certaine dimension. Lorsque l'angle de projection s'adapte bien à l'orientation du document, on peut alors repérer une différence importante entre deux valeurs successives dans la courbe d'intensité comme la limite entre deux blocs de texte. Le problème est l'estimation de différences significatives dans l'histogramme. En général, on utilise des connaissances a priori sur le document scanné (nombre de colonnes, marges, etc.) pour localiser plus facilement les séparations.

Nicolaou et Gatos [100] ont fait l'hypothèse que pour chaque ligne de texte, il existe un chemin d'un côté de l'image à l'autre qui ne traverse qu'une ligne de texte. Après avoir estompé l'image, en appliquant un filtre moyenneur, ils utilisent des traceurs de minima locaux pour suivre le chemin le plus blanc et le plus noir de gauche à droite ainsi que de droite à gauche afin de découper l'image en lignes de texte.

Ces méthodes de projections globales ne peuvent pas gérer les documents inclinés
Pour faire face à ce cas très fréquent, une solution consiste à rechercher d'abord l'orientation du document en recherchant des lignes droites comme le proposent Shapiro et al. dans [101]. L'angle d'orientation du document est d'abord estimé par transformée de Hough, puis utilisé pour calculer les projections. Le nombre de maxima du profil de projection donne le nombre de lignes. Les maxima locaux secondaires sont rejetés en comparant leurs valeurs avec celles des plus hauts sommets. Les résultats obtenus sont satisfaisants.

Toutes ces méthodes basées sur la détection de régions ou sur les projections ont permis d'atteindre de bonnes performances. Elles présentent cependant quelques inconvénients qui limitent leur usage. Par exemple, les mots constitués d'un seul caractère sont ignorés par les règles de regroupement par souci de précision, les caractères de faible contraste de couleur ne peuvent pas être extraits par MSER et le post-traitement est assez complexe.

3.4.2.c. Méthodes basées sur le Deep Learning

Comme on a pu le voir dans la section 3.2.7, les approches basées sur les CNNs ont permis d'atteindre des performances inédites en termes de détection d'objets (*R-CNN [46], Fast R-CNN [48]*. Inspirées par les progrès des CNNs en détection d'objets, plusieurs approches de détection de texte se basant sur le Deep Learning ont été envisagées [102, 103, 25, 104, 105, 106, 107].

Une des premières méthodes proposée faisant appel aux CNNs est celle de Wang et al [108]. Ils présentent une approche qui consiste à classifier des fenêtres glissantes (de taille 32 par 32 pixels, adaptée à une taille de caractères) à l'aide d'un CNN. Le réseau convolutionnel est composé de deux couches de convolutions, deux couches de max-agrégation et une couche complètement connectée. Il est utilisé ici simplement en tant que classifieur en deux classes (texte et non-texte).
Un autre groupe de méthodes est basé sur des réseaux dédiés à la segmentation sémantique d'images. Zhang et al. [105] utilisent le *Fully Convolutional Network* (FCN)

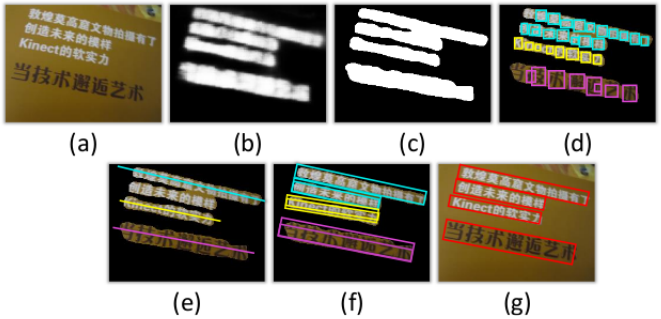


FIGURE 3.18. – Procédure de la méthode proposée par Zhang et al. : images extraites de [105].

[55] pour obtenir des cartes de segmentation qui segmente les zones de l'image qui ont la plus forte probabilité de contenir du texte. Le problème est que les zones candidates peuvent être connectées les unes aux autres, et leurs limites sont souvent floues, comme on peut le voir sur l'illustration 3.18 (c). Pour faire les prédictions finales sous forme de rectangle, par ligne, les auteurs doivent mettre en place des contraintes fortes de recalage de l'intensité et de la géométrie en considérant que les caractères d'une ligne de texte forment une ligne droite. Jaderberg et al. [25] propose également une méthode similaire à celle-ci. Il cherche à localiser les mots dans une image. Pour cela, il génère une carte de segmentation, de la même taille que l'image originale en évaluant les pixels de l'image au travers d'un CNN en tant que classifieur de 2 classes (caractères et arrière-plan). À partir de cette carte de segmentation, les lignes de textes sont localisées à l'aide de l'algorithme RLSA (*run length smoothing algorithm*). Les performances atteintes pour la détection des caractères restent intéressantes puisqu'ils atteignent une précision de 88% et un rappel de 78% sur une base de tests (ICDAR 2013) où l'état de l'art était à 89% pour la précision et seulement 70% pour le rappel.

Les approches basées sur le Deep Learning ont permis de surpasser les performances de l'état de l'art. Le principal atout de ces approches est leur capacité à généraliser sur des exemples qu'elles n'ont jamais rencontrés. En effet, nous avons testé le CNN proposé dans [25], entraîné sur des images de scènes naturelles, en lui soumettant une image contenant un ticket de caisse. Les performances obtenues étant très satisfaisantes. Nous avons continué de creuser la piste des réseaux de Deep Learning pour la détection des zones de caractères.

3.4.3. Reconnaissance des caractères

La reconnaissance optique de caractères consiste à identifier les caractères dans une image. Plus précisément c'est un processus de reconnaissance des caractères de l'image d'entrée pour les convertir en ASCII ou autre forme équivalente compréhensible par une machine. La tâche de reconnaissance a pour but d'extraire des imagettes dans l'image contenant un mot ou un caractère et de prédire l'information textuelle contenue dans

cette image. Les approches pour la reconnaissance de caractères peuvent être divisées en deux grands groupes : les approches basées sur la reconnaissance de caractères et celles basées sur la reconnaissance de mots. Pour ces différentes approches, nous allons présenter les méthodes traditionnelles, puis celles basées sur les réseaux de neurones profonds.

3.4.3.a. Méthodes « Traditionnelles »

La reconnaissance de caractères repose sur un classifieur dédié permettant de reconnaître individuellement chaque caractère contenu dans l'image. Traditionnellement, les différentes approches de reconnaissance de texte sont basées sur la classification séquentielle des caractères et se concentrent principalement sur le développement d'un classifieur de caractères puissant utilisant des caractéristiques images définies a priori « manuellement ». L'extraction d'images de caractères se fait, comme nous l'avons vu au paragraphe précédent, par des fenêtres glissantes, ou encore la recherche de composants connectés. Neumann et Matas [91] proposent une détection et une classification des caractères basées sur l'orientation des traits. Leurs performances sont limitées par les caractéristiques bas niveau utilisées. Yao et al. [109] proposent un modèle original de représentation (*Strokelets*) permettant de décrire des sous-patchs de caractères. Leur algorithme est à la fois efficace et robuste. Des expériences approfondies sur des benchmarks standards vérifient les avantages des strokelets et démontrent que l'algorithme atteint des performances intéressantes.

Ces techniques fonctionnent généralement bien sur les ensembles de données homogènes pour lesquels elles ont été optimisées, mais nécessitent une ingénierie lourde pour bien fonctionner sur des ensembles de données hétérogènes, c'est-à-dire que les modèles de représentation doivent être suffisamment représentatifs de l'ensemble de données. Pour cette raison, d'autres méthodes se sont imposées, en particulier les réseaux convolutionnels profonds qui se sont avérés être efficaces.

3.4.3.b. Méthodes basées sur des Réseaux de neurones

Reconnaissance de caractères Les récentes avancées des réseaux de neurones convolutionnels en matière d'analyse d'image ont permis de proposer des méthodes de reconnaissance/classification de caractère à haut niveau de performance, en termes de rapidité comme de précision. Les travaux [110, 111, 112] utilisent tous des réseaux convolutionnels profonds comme classifieurs de caractères. [110] et [111] segmentent l'image d'un mot en régions contenant potentiellement un caractère, soit par des techniques de binarisation (de l'image d'entrée) [111], soit par un classifieur supervisé [110]. Le système « Photo OCR », présenté par Bissaco et al. dans [111], utilise un réseau de neurones « classique » composé de cinq couches complètement connectées avec la fonction d'activation ReLu (cf. paragr. 3.2.3). Les auteurs ont exploré l'utilisation de réseaux de neurones convolutionnels, mais ont abouti à des architectures peu intéressantes en termes de coût

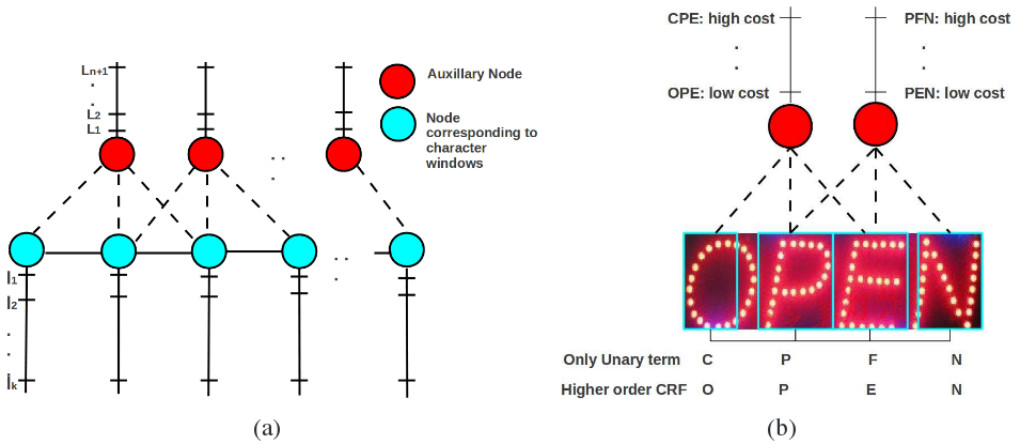


FIGURE 3.19. – Graph présenté par [113] permettant la reconnaissance de mot.

de calcul. Alsharif et al. [110] utilisent un réseau CNN de type *Maxout* composé de cinq couches de convolution, prenant en entrée des images de taille 32 par 32. Sur la base de ICDAR2003, ils atteignent une précision de 85,5%.

Wang et al. [112] utilise également un réseau de neurones contenant deux couches de convolutions avec respectivement $n_1 = 115$ et $n_2 = 720$ filtres, en tant que classifieurs de 62 classes (26 lettres majuscules, 26 lettres minuscules et 10 chiffres). Toujours sur la base ICDAR2003, ils obtiennent de meilleures performances avec une précision de 90%.

Les réseaux présentés sont dans l'ensemble peu profonds car les bases d'apprentissage sont de petite taille. La plupart des méthodes présentées ont recours à de l'augmentation de données en concaténant plusieurs bases de données où en générant des données synthétiques. Les résultats obtenus sont cependant intéressants.

Reconnaissance de mots En tant qu'approche alternative à la reconnaissance de caractères, d'autres méthodes utilisent la reconnaissance basée sur des mots entiers. Ces méthodes regroupent des caractéristiques de l'ensemble de la sous-image mot avant d'effectuer la classification des mots. Les travaux de Mishra et al. [113] et Novikova et al. [114] reposent encore sur des classificateurs de caractères explicites, mais construisent un graphique pour déduire le mot, regroupant les preuves de mots complets, comme illustré sur la Fig. 3.19. Goel et al. [115] construisent des images synthétiques en noir et blanc avec les différents mots d'un lexique avec différentes polices et différentes tailles. La reconnaissance des mots dans les images se fait tout simplement en faisant correspondre les caractéristiques de l'image requête et celle de l'image synthétique.

Jaderberg et al [116] prennent en entrée de leur système une image d'un unique mot, redimensionnée à une taille de 32 x 100 pixels. Le système basé sur un réseau de neurones convolutionnels classifie alors cette image comme un des mots d'un dictionnaire. Dans ce travail, le dictionnaire utilisé contient 90 000 mots anglais. Le réseau, illustré sur la fig. 3.20, est composé de cinq couches de convolution et trois couches complètement connectées, la dernière couche est celle qui réalise la classification à travers le diction-

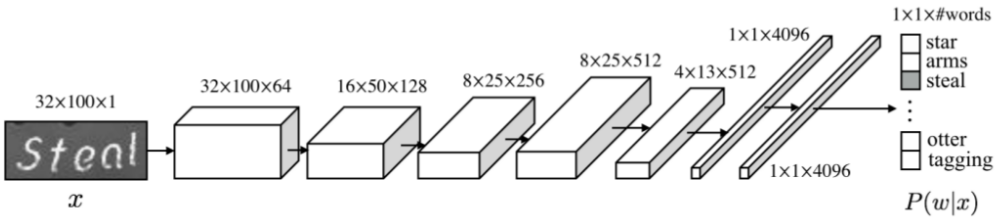


FIGURE 3.20. – Réseau CNN pour la reconnaissance de mots proposés par Jaderberg et al. : images extraites de [116].

naire de mots, elle a donc le même nombre d'unités que la taille du dictionnaire qu'ils souhaitent reconnaître, soit 90 000. Cependant, pour pouvoir entraîner un réseau aussi profond sur autant de classes (90 000), de nombreux échantillons de tous les différents mots possibles doivent être collectés. Un tel ensemble de données d'entraînement n'existant pas et une base de données d'entraînement synthétique a donc été utilisée. Ces données synthétiques sont si réalistes que le CNN, bien qu'entraîné uniquement sur des données synthétiques, peut être appliqué à des données réelles.

Dans notre situation, l'application de la reconnaissance de mots n'est pas applicable sur l'ensemble du ticket. En effet, la section contenant la liste des produits apparaît souvent sous forme de libellés courts non standardisés (n'appartenant à aucun dictionnaire). De plus, les enseignes refusent généralement de fournir la syntaxe et la définition de ces libellés. Il n'est donc pas envisageable de dresser un dictionnaire exhaustif des libellés courts existant dans le ticket de caisse. De plus, ce dictionnaire devrait être mis à jour en continu au rythme des disparitions et apparitions quotidiennes de produits.

3.4.4. Détection et Reconnaissance de caractères

Nous avons vu différentes approches permettant de réaliser d'une part la détection des zones de textes et d'autre part la reconnaissance de caractères.

Jaderberg et al. [25] présente une architecture de réseau de neurones convolutionnel permettant de réaliser à la fois de la détection de zone de textes et la reconnaissance de caractères. Ils montrent de manière empirique que l'utilisation de « *Maxout* » dans ce réseau, au lieu d'utiliser la fonction d'activation ReLU, amène à de meilleures performances. Ce CNN prend en entrée une image de taille 24 x 24 pixels, pour donner quatre sorties : un classifieur caractère/arrière-plan (2 classes), un classifieur de caractères non sensible à la casse (37 classes), un classifieur de caractères sensible à la casse (63 classes) et un classifieur de bigrammes (couple de caractères, 604 classes) . Ces quatre sous-réseaux partagent les deux premières couches qui permettent d'extraire des caractéristiques génériques des différents caractères et se terminent par deux couches spécialisées pour chacune des quatre tâches différentes. Pour entraîner un tel réseau, Jaderberg et al. l'ont fait de manière séquentielle en deux étapes. Ils commencent par

entraîner le réseau pour la tâche de classification de caractères non sensible à la casse. Puis, ils utilisent la sortie de la deuxième couche pour entraîner les trois autres classifieurs en figeant les paramètres des deux premières couches. Cette architecture de réseau est un réseau multi-tâche séquentiel. Ces réseaux auraient pu être entraînés de manière indépendante. Cependant, partager les deux premières couches a deux avantages clés. Premièrement, les fonctions de bas niveau apprises à partir de la classification des caractères insensible à la casse permettent de partager les données d'entraînements entre les tâches, réduisant ainsi le sur-apprentissage et améliorant les performances pour les tâches de classification avec moins de données d'apprentissages (classification des caractères sensibles à la casse, classification des bigrammes). Deuxièmement, cela permet de partager des calculs, ce qui augmente considérablement l'efficacité. Ils obtiennent de très bonnes performances, sur la base ICDAR2003. En ce qui concerne la reconnaissance de mots sur une image croppée ils atteignent une précision de 91,5%, ce qui est meilleur que la proposition de Alsharif et al. [94]. Quant au système complet c'est-à-dire la détection et la reconnaissance ils obtiennent une F-mesure de 80%.

Encore une fois, cette approche permet de surpasser l'état de l'art.

Un autre exemple dans la littérature, où la détection de zones de texte et la reconnaissance de caractères sont associées est le travail de travaux de Moysset et al. [117]. La proposition avancée consiste à transférer une partie de l'étape difficile qu'est la localisation à la reconnaissance. L'étape de localisation est en charge de détecter uniquement les départs des lignes de texte. L'étape de reconnaissance se charge de trouver où la ligne de texte s'arrête. Chaque étape met en jeu un réseau de neurones convolutif spécifique. Le premier CNN qui permet de localiser le début d'une ligne de texte est un réseau purement convolutionnel (il n'y a pas de couches complètement connectées) avec cinq couches de convolutions. Le réseau prédit trois valeurs : les coordonnées pour le coin inférieur gauche, la hauteur du texte et l'indice de confiance. Une fois le début de la ligne détectée, la localisation du bloc de texte est décalée de manière itérative vers la droite. Le deuxième CNN, entraîné à reconnaître les caractères, mais également à prédire la fin d'une ligne en lui assignant un label « EOL » (*End of Line*), termine la tâche de détection et de reconnaissance de texte, comme illustré sur la Fig. 3.21.

Solutions commerciales existantes Comme nous avons pu le voir au chapitre 2, la Reconnaissance Optique des Caractères (OCR) est un domaine tellement actif et demandé que plusieurs solutions existent aujourd'hui telles que Tesseract [19] ou encore plus récemment Google Vision [21] qui est basé sur des réseaux de neurones profonds ce qui a permis d'améliorer encore les performances. Ces solutions commerciales réalisent à la fois de la détection et la reconnaissance de caractères. L'API Google Vision est un système relativement complexe qui permet aux développeurs de comprendre le contenu d'une image en encapsulant de puissants modèles d'apprentissage, facile à utiliser. Il classe rapidement les images en milliers de catégories, détecte les objets et les visages in-

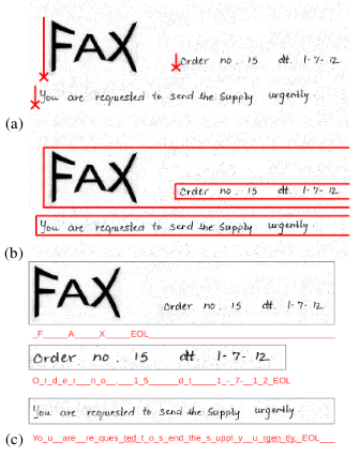


FIGURE 3.21. – Détection et Reconnaissance de texte dans un document, images extraites de [117].

dividuels dans les images, et recherche et lit les mots imprimés contenus dans les images. L'inconvénient majeur de ces solutions est qu'elles sont des boîtes noires dont nous ne maîtrisons pas tous les paramètres.

3.5. Analyse Sémantique

3.5.1. Introduction

De nombreuses applications du Traitement Automatique des Langues Naturelles, comme l'indexation de textes, la traduction automatique, ou encore le résumé automatique, sont potentiellement demandeuses d'une analyse sémantique aussi fine que possible des textes. Il peut s'agir par exemple d'extraire la thématique générale d'un document sous la forme de sélection de concepts prédéterminés. De plus, avec la masse d'informations aujourd'hui disponible via le Web, en évolution permanente, il est devenu nécessaire de savoir définir automatiquement une structuration afin de faciliter l'accès et la gestion des connaissances.

L'analyse sémantique d'un texte est une branche linguistique et peut être définie comme une tâche visant à effectuer un certain nombre de traitements dans le but d'établir la signification en utilisant le sens des éléments (mots) du texte, dans leur contexte. Par opposition aux analyses lexicales ou grammaticales qui décomposent le message à l'aide d'un lexique ou d'une grammaire, c'est une phase très complexe qui fait généralement appel à de gigantesques bases de connaissance, comme les thésaurus. Un thésaurus est une liste organisée de termes contrôlés et normalisés représentant les concepts d'un domaine de connaissance. C'est une forme d'ontologie.

Dans nos travaux, l'analyse sémantique est utilisée comme un outil, au travers des

techniques restant à un niveau relativement simple, telle que les expressions régulières permettant d'extraire de la donnée structurée, la construction de bases de connaissance et de la définition de terminologies, mais aussi la construction d'une ontologie, permettant l'interprétation du contenu.

Nous donnons cependant dans cette section un aperçu plus général de l'état de l'art dans le domaine de l'analyse sémantique de manière à montrer les liens potentiels avec les travaux de cette thèse. C'est sur cet état de l'art que pourra s'appuyer la suite de ces travaux de thèse.

3.5.2. Approches de l'analyse sémantique des textes

L'analyse sémantique des textes peut être abordée de deux manières : une méthode non supervisée ou supervisée [118]. La méthode non supervisée vise à découvrir le contenu d'un texte sans a priori. La méthode supervisée s'applique dans le cas de textes appartenant à un domaine particulier, pour lesquels l'analyse tend à rechercher des informations spécifiques [119]. Nous nous intéressons, ici au cas d'une analyse supervisée.

Dans la littérature, plusieurs méthodes permettent d'appréhender l'analyse sémantique de manière supervisée. Dans ce cas-là, il est nécessaire de définir des règles pouvant être modélisées. Or plusieurs modélisations sont possibles et deux d'entre elles, les plus courantes, retiennent notre attention : il s'agit des ontologies et de la terminologie. Deux termes issus de deux domaines d'expertises très différents, mais qui sont pourtant de plus en plus associés [120].

Effectivement, les règles définies, afin d'appréhender l'analyse sémantique d'un texte, peuvent être représentées sous forme d'une ontologie. Les ontologies ont pour objectif premier de modéliser l'ensemble des connaissances au sujet d'un monde ou d'une partie de ce monde et elles sont souvent employées dans l'intelligence artificielle. Elles permettent de modéliser de manière structurée les concepts d'un domaine ainsi que les relations entre ses concepts, comme illustrées sur la Fig.3.22. Les méthodes de construction d'ontologies à partir de textes comportent donc une phase de conceptualisation [121] au cours de laquelle s'effectue le passage du terme au concept. Pinto et al. [122] montrent que l'utilisation d'une ontologie pour soutenir l'analyse approfondie des bases de données marketing permet de surpasser les problèmes de complexité et de quantité de données qui limitent l'application des autres techniques d'analyse comme celle nommée "la découverte des connaissances" [123].

Cependant, cette modélisation conceptuelle du domaine n'est pas suffisante dans le cas de corpus contenant une diversité linguistique éloignée de la langue actuelle, composée d'acronymes et de termes polysémiques, très différents selon les sources des textes. La terminologie qui dénote les concepts du domaine doit être prise en compte, comme le propose Schuster et al. [125]. Une terminologie permet de définir l'ensemble des termes qui sont spécifiques d'une science, d'un domaine particulier de l'activité humaine. Dans

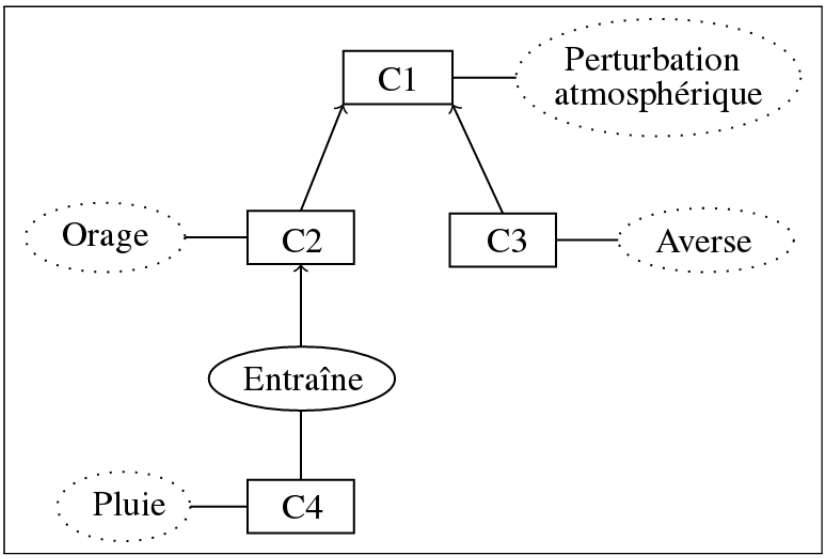


FIGURE 3.22. – Illustration d'un exemple d'une ontologie extrait de [124].

[125], les auteurs présentent une méthode permettant d'intégrer la terminologie tout en se basant sur l'ontologie. Cette intégration sémantique s'appuie sur l'idée que chaque source d'information sert de contexte pour l'interprétation de l'information qui y est contenue. Ainsi une information ne peut être complètement comprise que si les informations contextuelles sont préservées.

Selon les expériences des équipes de recherche en ingénierie de l'ontologie, la construction d'ontologies est beaucoup plus longue et coûteuse que la construction de terminologies en termes de complexité et de formalité des ontologies. D'autant plus que la pertinence des informations que les ontologies contiennent nécessite une mise à jour régulière. C'est pourquoi de nombreux travaux proposent des approches d'enrichissement d'ontologie [124, 126, 127]. Le processus d'enrichissement d'ontologie peut être divisé en deux étapes : la recherche de nouveaux concepts et relations et le placement de ces concepts et relations au sein de l'ontologie. Par exemple Di Jorio et Al. proposent une méthode d'enrichissement automatique d'ontologie basée sur des techniques de fouille de données (recherche de motifs séquentiels) dans des documents textuels. Leur approche a été testée sur une ontologie du domaine de l'eau et a été enrichie à partir de documents issus du Web.

L'enrichissement d'une ontologie de manière automatique est encore un sujet de recherche actuel. C'est une tâche difficile, puisqu'il s'agit d'établir et de nommer des relations entre concepts selon une vision d'un monde. Il n'y a pas qu'une seule manière, ni une manière meilleure que les autres de décrire une partie de la réalité.

3.5.3. Compréhension du texte

La compréhension du texte consiste à lire des textes formés dans des langages naturels, à déterminer la signification explicite ou implicite de chaque élément comme des mots, des phrases, des paragraphes. Ce problème est difficile en raison de la variabilité des langues.

3.5.3.a. Méthode traditionnelle

Jusqu'à présent, la plupart des approches d'apprentissage automatique de la compréhension de texte nécessitent de larges connaissances a priori. Elles doivent prédéfinir un dictionnaire de mots, et un analyseur syntaxique. Ces exigences rendent la compréhension du texte plus ou moins spécialisée à une langue particulière, si la langue est modifiée, beaucoup de choses doivent être repensées.

3.5.3.b. Deep Learning et Analyse sémantique

Avec l'avancement du Deep Learning et la disponibilité de grands ensembles de données, les méthodes de la compréhension du texte en utilisant des techniques Deep Learning sont devenues progressivement disponibles. « Word2vec » est une technique très utilisée [128]. Cette technique construit la représentation des mots dans un vecteur de longueur fixe, formé sous un grand corpus. S'appuyant sur l'espoir que les machines puissent donner un sens aux textes, de nombreux chercheurs ont essayé de former un réseau de neurones pour comprendre les textes basés sur les caractéristiques extraites par la technique « Word2vec » ou des techniques similaires [129, 130].

Les auteurs de [131] montrent qu'un système de Deep Learning permet de comprendre un texte sans apporter aucune connaissance a priori sur les mots, phrases où encore la syntaxe. Le réseau utilisé est composé de 9 couches : 6 couches de convolutions et 3 couches complètement connectées. L'évaluation de cette approche sur différentes bases de données, dont une en chinois, permet de surpasser les résultats de l'état de l'art et prouve qu'il est facilement adaptable à différente langue.

3.5.4. Apports de l'analyse sémantique

L'analyse sémantique dans notre cas peut intervenir à différentes étapes et permettre d'évaluer la performance des étapes précédentes. Dans un premier temps, elle permettra de corriger les éventuelles erreurs de la reconnaissance de caractères (OCR, *Optical Character Recognition*). Ensuite, elle permettra d'extraire les différentes informations présentes dans le ticket de caisse. Mais surtout, l'utilité est de pouvoir retranscrire les libellés des produits achetés. En effet, comme exposé au chapitre 2, par faute de place, les libellés des produits apparaissent généralement sous forme de libellés courts. Sachant qu'il n'existe aucune règle standard qui permet de passer du libellé court au libellé complet d'un produit. Par exemple, le libellé « *pdm sdw cereal* » signifie « *Pain de mie sandwich aux céréales* ». Une autre difficulté est de prêter attention au contexte

qui importe beaucoup, puisqu'un même abrégé peut signifier des choses complètement différentes, c'est là que vont intervenir l'ontologie et la terminologie. Prenons comme exemple, l'abrégé « *pdt* » qui peut fait référence à « *Pomme de terre* » mais aussi à la marque « *President* » (à relier au produit acheté « *Beurre President* »).

Notre objectif est d'analyser des données textuelles de consommation provenant de sources différentes, donc dans une « langue » différente : « langue des marques », « langue de chaque supermarché ». En effet, chaque magasin possède ses propres termes pour définir un produit donné. C'est pourquoi les outils de l'analyse sémantique tels que les ontologies et la terminologie permettront d'instaurer un langage pivot qui permettra de faire le lien entre les différentes langues. Par exemple, le papier toilette peut être défini sous le terme « papier hygiénique », « papier WC », ou encore le libellé court « pH ». Afin d'être en mesure de connaître la consommation de ce produit de manière exhaustive, il est nécessaire d'utiliser une modélisation conceptuelle de l'objet et de lui associer sa terminologie. Dans l'idée qu'une ontologie est un outil utilisable aussi bien par l'homme que par les machines , elle permet d'assurer l'usage d'un vocabulaire commun et se mettre d'accord sur le sens des termes employés dans une communauté.

3.6. Bilan

Au paragraphe 3.2, nous avons détaillé les différentes approches existantes basées sur le deep learning permettant de faire des détections d'objets et des classifications d'images. Dans nos travaux, différents objets sont à détecter comme le ticket, le logo et les différents blocs de textes.
Nous avons alors présenté des différentes méthodes plus spécifiques pour la détection et la classification des logos dans le paragraphe 3.3 et la détection et la reconnaissance de textes dans le paragraphe 3.4. Parmi toutes les approches présentées pour les différents cas, le Deep Learning apparaît systématiquement et permet de surpasser l'état de l'art. Le Deep Learning nous intéresse donc dans toutes nos réalisations. L'analyse sémantique des textes présentée au paragraphe 3.5 est l'approche sur laquelle nous nous focalisons pour l'interprétation du contenu extrait de l'image analysée.

4. Pré-traitements

Dans ce chapitre, nous nous intéressons à la phase de pré-traitement de l'analyse automatique du contenu d'un ticket à partir d'une image prise par un terminal mobile. Dans un premier temps, nous commençons par rappeler les différentes étapes envisagées (détection du ticket, localisation du ticket et détection de l'enseigne) tout en expliquant pour chacune les objectifs fixés. Puis, dans un second temps nous présentons les méthodes développées et les performances obtenues.

4.1. Introduction

Nous avons proposé à la fin du chapitre 2 , au paragraphe 2.4 , une chaîne de traitement permettant de répondre aux fortes contraintes industrielles concernant notamment la fiabilité de chaque information extraite (cf schéma 2.16 au chapitre 2). L'expérience acquise lors de l'évaluation du démonstrateur nous a permis de définir une chaîne de traitement composée de six étapes, dont plusieurs étapes concernent uniquement la préparation de l'image afin d'assurer une lecture dans des conditions favorables. En effet, cette chaîne de traitement peut être divisée en deux parties : une partie « pré-traitement » et une partie « Lecture et Analyse du contenu ». La partie « pré-traitement » consiste à obtenir une image d'un ticket de caisse la plus propre possible isolée et débarrassée du contexte d'acquisition. La deuxième partie « Lecture et Analyse du contenu » concerne le cœur du projet autrement dit la détection des blocs de texte, la lecture et l'analyse de l'information textuelle du ticket. Dans ce chapitre, nous nous intéressons à la partie pré-traitement qui comme nous avons pu le constater est une étape importante pour assurer une bonne qualité d'analyse du ticket.

D'un point de vue technique, l'étape de pré-traitement a plusieurs objectifs. Dans un premier temps, elle doit permettre d'assurer que l'image en entrée contient un ticket de caisse analysable. Ensuite cette étape permet d'extraire les différentes informations intéressantes et nécessaires à la compréhension du contenu du ticket et d'éliminer les informations sans intérêts qui peuvent être perturbatrices (fond de l'image).

Parallèlement, cette phase de pré-traitement présente également un intérêt commercial. En effet, afficher rapidement différents résultats du pré-traitement sur le mobile de l'utilisateur montre que l'image acquise a bien été prise en compte et que son analyse est lancée. Par ces retours vers l'utilisateur nous favorisons l'adhésion à l'application. Cet affichage ne doit pas forcément se faire en temps réel et peut-être reporté de quelques secondes (1 à 2 secondes) après la prise de photo. Ainsi l'utilisateur peut se rendre compte assez rapidement de « l'intelligence » du système, ce qui est un critère important pour

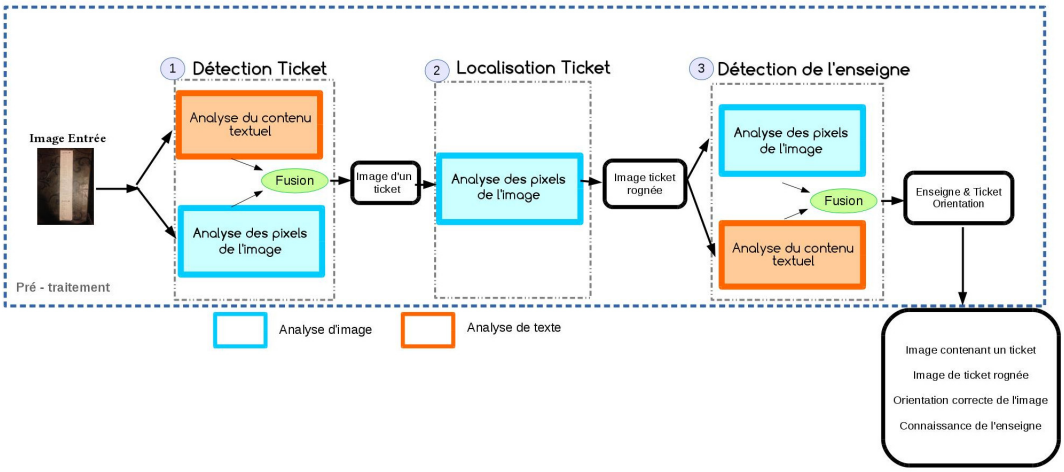


FIGURE 4.1. – Chaîne de traitement globale pour le pré-traitement d'analyse d'un ticket.

l'image de l'entreprise et de ses produits. En revanche, un ressenti émotionnel globalement positif de l'utilisateur face à l'interface reste conditionné par un taux de confiance du processus suffisamment élevé.

La chaîne de traitement est mise en place afin de répondre aussi bien aux intérêts techniques que commerciaux. Elle est donc composée de trois étapes comme illustrée sur la figure 4.1 qui sont construites de façon à assurer une chaîne de traitement globale avec peu de fausses alarmes. Il est en effet préférable que le système ne donne pas l'information recherchée plutôt qu'il donne une fausse information. Le système doit donc être capable d'évaluer son degré de certitude pour chaque information extraite. Cette chaîne de traitement doit également permettre d'apporter un certain nombre d'informations à l'utilisateur avant la lecture complète du ticket et tout cela, en un temps de calcul réduit. Il s'agit donc de trouver le bon compromis entre le temps de calcul et la performance du système.

La première étape est la détection du ticket dans l'image. Considérant une acquisition « libre », tout type d'image peut être amené à être traité. Un échantillon illustre cette variété d'images dans la figure 4.2 .
Deux analyses différentes, une analyse des pixels de l'image et une analyse plus spécifique du contenu textuel sont menées parallèlement. La décision finale est prise en fusionnant les deux résultats.

La deuxième étape consiste à localiser de façon précise le ticket de caisse afin d'obtenir une image détourée du ticket autrement dit une image débarrassée de son contexte d'acquisition. Cette étape permet de faciliter la phase suivante du traitement, « Lecture et Analyse du contenu » car elle permet d'une part d'assurer que l'image analysée ne

FIGURE 4.2. – Échantillon d'images reçues en entrée du système.

contient que du texte et d'autre part de corriger l'éventuelle non verticalité du ticket, critère important pour sa lecture. Toujours dans le but d'atteindre de bonnes performances, deux traitements différents seront appliqués de manière séquentielle.

La troisième étape est la détection de l'enseigne. Cette étape a pour but d'apporter un maximum de connaissance a priori sur la structure du ticket avant de commencer sa lecture. En effet, chaque enseigne a une manière spécifique de présenter un ticket de caisse, des polices et tailles de police propres, des espacements particuliers entre lignes, entre caractères... Comme pour les deux étapes précédentes, nous allons assurer la détection de l'enseigne en parallélisant deux analyses (une analyse d'image et une analyse du texte contenu dans le ticket).

Dans la suite ce chapitre nous allons étudier et analyser les différents modules de l'étape de pré-traitement. Dans une première section, nous présenterons des éléments communs aux différentes étapes. Puis, les sections suivantes décriront successivement chacune des trois étapes en détaillant à chaque fois la méthodologie puis les résultats obtenus. Enfin, nous terminerons par une conclusion.

4.2. Préambules

L'expérience acquise lors de l'évaluation du démonstrateur au chapitre 2, appuyée par l'étude de l'état de l'art sur la localisation et la détection d'objets dans des images (chapitre 3), nous a naturellement amenée à nous intéresser aux méthodes basées sur le Deep Learning, afin de répondre aux objectifs des trois tâches de la phase de pré-

traitement. Nous allons maintenant expliquer la manière dont nous avons utilisé ce type d'approche.

4.2.1. Les Architectures DL de classification

4.2.1.a. Choix de la méthode d'apprentissage

Le Deep Learning ici, doit nous permettre de répondre à deux tâches différentes :
— la localisation d'objets, à savoir la localisation du ticket (étape 2) et la localisation du logo (étape 3). Les réseaux mis en jeu ici sont donc des réseaux de segmentation sémantique ;
— la classification d'objets (logo, ticket), qui , bien sûr, utilisera des réseaux classifieurs.

Une forte contrainte concernant la faible quantité de données annotées à notre disposition (voir section 4.2.3) nous impose le choix de la méthode d'apprentissage. En effet, il n'est pas envisageable dans notre situation de construire un nouveau réseau de neurones profond et de l'optimiser uniquement sur le peu de données disponibles. C'est pourquoi nous avons fait le choix de nous appuyer sur des réseaux existants ayant déjà fait leur preuve avec des stratégies de « transfer learning » et éventuellement de « fine tuning » (voir section 3.2.6 au chapitre 3).

4.2.1.b. Choix des architectures de DL utilisées

Les catégories à reconnaître (ticket, logos) ne sont pas trop complexes. Ce sont en effet des cibles planes dont la variabilité est bien plus faible que des classes d'objets 3D à forte variabilité intra et inter classes comme celles que l'on peut trouver dans les grandes bases de données de référence du domaine comme ImageNet.
Pour des raisons liées à la facilité de mise en place, notre choix s'est porté sur deux réseaux bien connus : Alexnet [31] et GoogLeNet [37], qui ont obtenu des résultats remarquables lors du challenge ILSVRC (voir chapitre 3 section 3.2.5).

4.2.1.c. Description des différentes configurations testées

Les architectures de bases et la méthode d'apprentissage étant définies, l'étape suivante consiste à définir précisément les couches à modifier et la manière de les modifier (« transfer learning » avec recours ou non au « fine tuning »).

Nous avons testé plusieurs configurations pour les architecture AlexNet [31] et GoogLeNet [37] pré-entrainées sur le challenge ILSVRC2012 .

AlexNet : Pour rappel, le réseau AlexNet est composé de cinq couches de convolutions (« conv1 », ..., « conv5 ») et trois couches complètement connectées (« fc6 »,« fc7 », « fc8 »), comme illustré sur la figure 4.3.

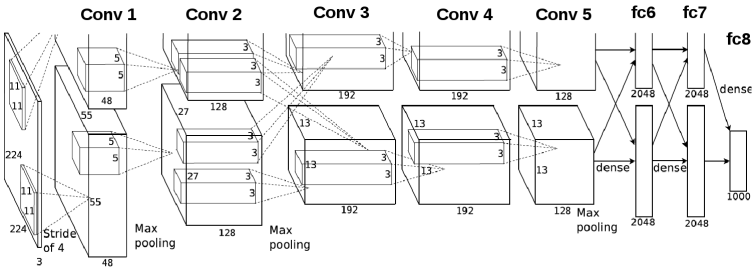


FIGURE 4.3. – Architecture du réseau AlexNet extraite de [31].

En accord avec les travaux de la littérature évoqués au chapitre 3, nous avons donc testé deux configurations différentes :

— transfert de toutes les couches du réseau de neurones initial sauf la dernière couche complètement connectée remplacée par une nouvelle couche, adaptée à la nouvelle tâche, qui sera la seule sujette à un entraînement. Cette architecture est nommée « AlexNet_lastFC » ;

— un transfert identique à la configuration précédente mais en introduisant cette fois un « Fine tuning » des deux premières couches complètement connectées (« fc6 »,« fc7 »), afin de leur permettre de s'adapter avec la nouvelle couche (« fc8 ») à la nouvelle tâche. Cette architecture est nommée « AlexNet_3FC ».

Dans la première configuration, seuls les neurones de la dernière couche « fc8 » sont modifiés. Pour cela, nous procédons en trois temps :

— ajustement du nombre de neurones de la couche « fc8 » au nombre de sorties de la nouvelle tâche, c'est-à-dire au nombre de classes du nouveau problème ;

— initialisation aléatoire des poids de ces neurones, suivant une distribution gaussienne. Tous les autres paramètres du réseau sont figés. Il s'agit donc des poids paramétrés pour répondre au Challenge ILSVRC2012 basé sur la base de données ImageNet ;

— ajustement des poids de la dernière couche grâce à une étape d'apprentissage utilisant les images d'entraînement spécifiques à la tâche considérée.

Dans la deuxième configuration, les couches « fc6 » et « fc7 » sont également modifiées, par la méthode de « finetuning ». Le processus est le suivant :

— comme précédemment, la couche « fc8 » est ajustée au problème considéré, les structures (nombre de neurones) des couches « fc6 » et « fc7 » restant inchangées,

— l'initialisation des couches « fc6 » et « fc7 » est celle du réseau AlexNet initial, alors que les poids de la nouvelle couche « fc8 » sont initialisés de manière aléatoire en suivant une distribution gaussienne ;

— les poids des trois couches sont ajustés (« fine tuning ») grâce à une étape d'entraînement avec des images annotées spécifiques à la tâche considérée.

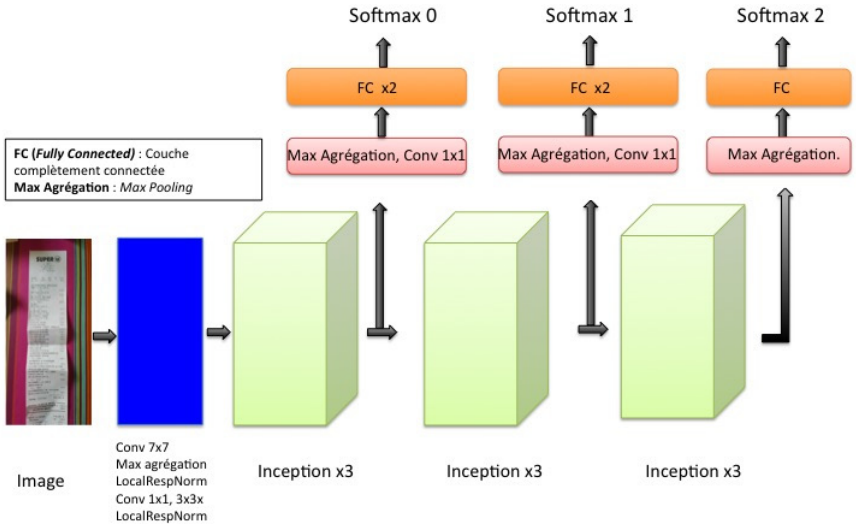


FIGURE 4.4. – GoogLeNet [37] est presque entièrement défini de modules d'Inception.

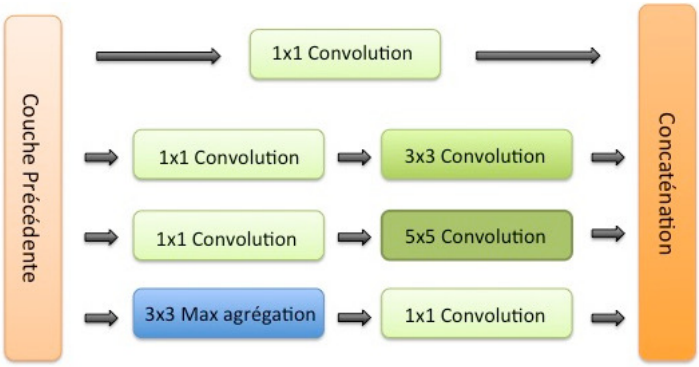


FIGURE 4.5. – Un module d'Inception défini dans GoogLeNet [37].

GoogLeNet : L'architecture du réseau GoogLeNet est plus profonde que celle d'AlexNet puisqu'elle contient 22 couches (voir section 3.2.5 du chapitre 3). Ce réseau introduit un nouvel élément nommé le module d'« Inception ». Ces modules, au nombre de trois dans GoogLeNet (figure 4.4), contiennent plusieurs résolutions de filtre de convolution (figure 4.5)

Après chaque module d'Inception (voir figure 4.4), se trouve un classifieur composé d'une couche de « max agrégation », de couches complètement connectées et un opérateur « softmax ». Ces classifieurs intermédiaires permettent d'atténuer le phénomène de la perte de l'influence des gradients (ou « *gradient vanishing* »), ces gradients permettant d'optimiser les paramètres du réseau.

Avec GoogLeNet, nous avons également testé deux configurations différentes :
— transfert de toutes les couches du réseau de neurones initial sauf la dernière couche

complètement connectée des trois classifieurs. Cette architecture est nommée « GoogLeNet_3Classifer » ;
— suppression des couches de classification intermédiaires, et transfert d'apprentissage de toutes les autres couches du réseau sauf de la dernière couche complètement connectée du dernier classifieur de l'architecture. Cette architecture est nommée « GoogLeNet_LastClassifer ».

Pour les deux configurations testées, le mécanisme mis en place est similaire à celui utilisé pour « AlexNet_lastFC » :
— ajustement du nombre de neurones au nombre de sorties de la nouvelle tâche ;
— initialisation aléatoire des poids des nouvelles couches toujours en suivant une distribution gaussienne ;
— ajustement des poids des nouvelles couches à l'aide d'une phase d'apprentissage en utilisant les images d'entraînement spécifiques à la tâche considérée.

Nous détaillerons les paramètres d'apprentissage utilisés pour chacune des étapes aux paragraphes correspondants (4.3.4 et 4.5.1).

4.2.2. Les Architectures DL de segmentation sémantique

L'un des objectifs de cette phase de pré-traitement est la détection de différents objets : le ticket de caisse et le logo. Le but recherché est de déterminer une boîte englobante autour des objets d'intérêt dont la précision reste grossière, mais suffisante. Elle se base sur le calcul d'une segmentation sémantique de l'image à une résolution faible. Cette méthode est plus rapide que des approches capables de chercher une estimation de la résolution initiale de l'image comme celles utilisées au chapitre 5 de ce manuscrit.

Pour répondre à ce problème de détection d'objets, nous nous sommes inspirée de la méthode proposée dans [44] et que nous avons décrite au chapitre 3 section 3.2.7.a qui consiste à transformer un réseau de classification en un réseau complètement convolutionel. Cette méthode permet de parcourir l'ensemble de l'image avec un pas correspondant au champ réceptif du réseau qui est de 32x32 pixels. Le résultat est un ensemble d'images sous-échantillonnées (une pour chaque classe cible), chaque image étant une carte de probabilité d'appartenance à une classe cible, comme illustré sur la figure 4.6. Par exemple, on obtient une carte de classification de dimension 32×32 sur une image en entrée de taille 1219×1219. Il y a eu donc 1024 classifications de petite image (de taille « 227 x 227 » pixels), mais contrairement à une méthode classique, ici le calcul exploite une efficacité naturelle de la structure convolutionnelle du réseau en amortissant le calcul des champs récepteurs qui se chevauchent.

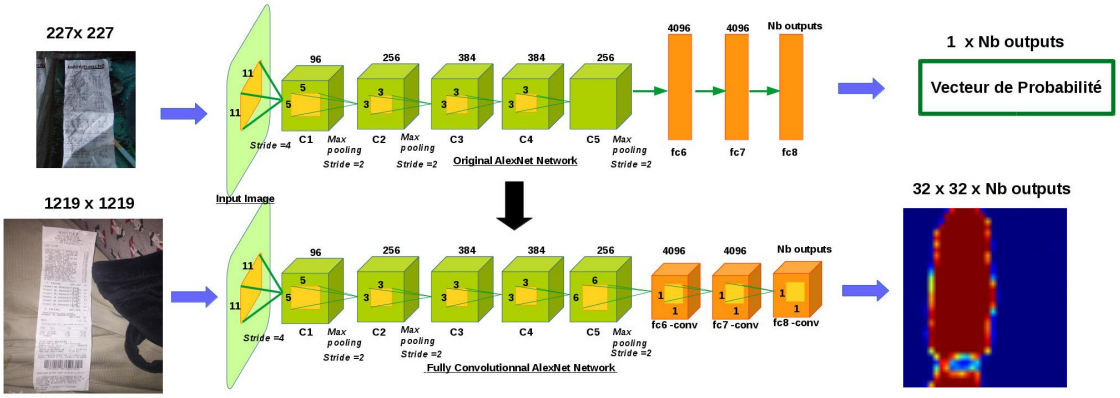


FIGURE 4.6. – La transformation d'un réseau de classification en un réseau entièrement convolutionnel permet d'obtenir une carte de probabilité en sortie, interprétée comme une segmentation sémantique. Schéma s'appuyant sur l'architecture AlexNet.

4.2.3. Obtention d'une base de vérité terrain

Étant donné, qu'à notre connaissance, aucune base de photos de tickets de caisses publiques n'existe, nous avons été amenée à constituer ces bases par nous même. Plus précisément, nous nous sommes donc concentrée sur la réalisation de deux bases : une première, pour la reconnaissance de tickets, et une deuxième, pour la reconnaissance de logos.

4.2.3.a. Récupération d'images

L'application « TachetKoa ? » (présentée au chapitre 2) proposée par l'entreprise et mise à disposition du grand public nous a permis de récupérer un certain nombre de photos prises par les utilisateurs. En effet, s'inspirant largement des meilleures applications proposant la même fonctionnalité, elle permet à l'utilisateur de prendre une photo sans trop de contraintes, comme illustrée sur la figure 4.7.

L'utilisation de ces images nous permet de nous placer dans les conditions réelles d'analyses et de pouvoir évaluer le système proposé sur une grande variété de situations offrant des qualités d'acquisition très variées (variété d'appareils utilisés, des conditions d'acquisition).

Nous avons également collecté des images de tickets issus des bases de données de certains clients d'AboutGoods. Les images fournies par les clients proviennent également de photos prises par des terminaux mobiles, en acquisition libre, mais aussi d'images scannées. La qualité des images à traiter varie donc énormément. Cette application a permis de collecter quelques dizaines de milliers d'images utilisées dans le cadre de cette thèse pour l'entraînement et la validation de la chaîne de traitement.

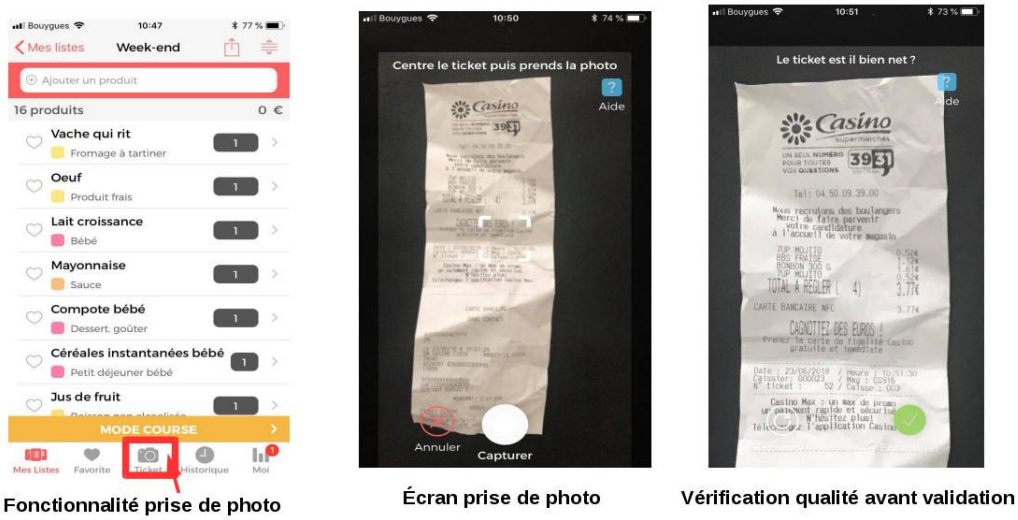


FIGURE 4.7. – Fonctionnalité d'acquisition d'image de ticket de caisse sur l'application TachetKoa ? proposée par AboutGoods Company.

4.2.3.b. Annotation

Parmi toutes les images à notre disposition, quelques dizaines de milliers d'images, il a fallu effectuer une première sélection en ne gardant que celles dont la qualité était acceptable, en terme de résolution, de netteté, etc. Toute image non humainement lisible, même en zoomant sur le ticket de caisse, a été rejetée.
L'annotation de toutes ces images, afin de répondre à nos deux problèmes de classification, nécessite la définition de la boîte englobant le ticket de caisse ou le logo dans le ticket de caisse. Cette boîte englobante est donc de forme rectangulaire et elle est décrite par les coordonnées de ses quatre sommets.

Une plateforme Web a été développée, pour faciliter et homogénéiser la réalisation de cette tâche. La plateforme est très simple d'utilisation, il suffit de sélectionner le type d'objet dont on souhaite définir la localisation (Ticket ou Logo), puis définir le rectangle englobant dans l'image directement (à l'aide de la souris) comme illustré sur la figure 4.8. Les coordonnées sont automatiquement sauvegardées en base de données.

Ainsi nous avons constitué une base d'apprentissage pour la reconnaissance de tickets de caisse contenant 3500 images : 2400 images contenant un ticket de caisse et la définition précise du rectangle englobant, 1100 images ne contenant pas de tickets.

En ce qui concerne la reconnaissance de logos, les logos étant très peu variables d'un ticket à un autre nous avons fait le choix de n'utiliser que très peu d'exemples par enseigne, à savoir 90 logos par enseigne. Cette décision a été imposée par le manque d'une variété suffisamment importante d'exemples de tickets de caisse pour certaines en-

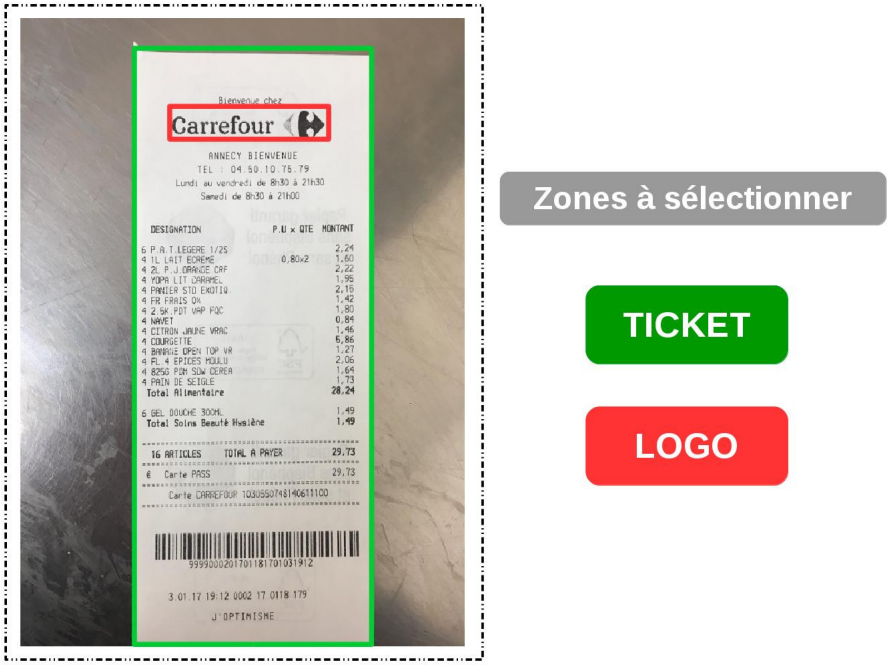


FIGURE 4.8. – Représentation de l'interface permettant l'annotation des tickets pour la détection des tickets et des logos.

Base de 17 Logos
Carrefour, Carrefour City, Carrefour Contact, Carrefour Market, Casino Supermarché, Cora, Géant, Grand Frais, Hyper U, Intermarché, Leclerc, Lidl, Match, Migros, Monoprix, Super U.

TABLE 4.1. – Liste des différentes enseignes de la base d'apprentissage

seignes. En effet, nous disposons de beaucoup d'exemples pour certaines enseignes telles que « Carrefour », ou « Casino Supermarché » et très peu pour d'autres enseignes. Nous avons donc dans un premier temps constitué une base de 10 enseignes, enrichie ensuite à 17 enseignes quand les images recueillies nous l'ont permis (voir tableau 4.1). Des échantillons des différents logos sont illustrés en Annexe.

Nous allons dans la suite de ce chapitre présenter les méthodes mises en œuvre et les performances obtenues pour chacune des trois étapes de la phase de pré-traitement, à savoir :

— détection du ticket ;
— localisation du ticket ;
— détection et localisation du logo.

4.3. Détection du ticket

Comme nous l'avons vu, la première étape du pré-traitement a pour but de pouvoir vérifier de manière automatique la présence ou non d'un ticket dans l'image à analyser. Comme nous pouvons le voir au chapitre 2, lors de cette première étape, deux approches différentes sont considérées afin de limiter les faux négatifs et d'assurer la fiabilité du traitement automatique. La première approche consiste en la compréhension et l'analyse des pixels de l'image, alors que la deuxième approche concerne l'analyse du contenu textuel de l'image.

4.3.1. Méthode proposée pour l'analyse d'image

Concernant l'analyse d'image, il s'agit d'un simple problème de classification à deux classes : Ticket/ Non ticket. Cependant, comme nous considérons une acquisition libre les tickets dans les images reçues peuvent être enfouis dans l'image avec un fond pouvant contenir beaucoup d'informations comme on peut le voir sur l'illustration 4.2. Cet aspect peut fortement nuire à la classification de ces images, et le risque est de manquer beaucoup de tickets (ce qui n'est pas envisageable pour l'entreprise). La solution proposée est d'utiliser un réseau de neurones profond, comme nous l'avons discuté plus tôt (cf 4.2.2), réalisant une segmentation sémantique par patchs et non une segmentation sémantique par pixel, afin de localiser les zones de l'image correspondant à un ticket. Pour cela, nous entraînons un réseau de classification afin de le transformer en réseau de segmentation sémantique (par patchs).

Entraînement du réseau de segmentation sémantique par patchs : Pour répondre à ce problème de classification, nous avons comparé les performances des quatre architectures de CNN présentées à la section 4.2.1.c. Concernant la phase d'entraînement, nous disposons de la base d'apprentissage que nous avons manuellement annotée pour la reconnaissance de tickets (section 4.2.3).

Comme l'objectif est d'apprendre un réseau réalisant une segmentation sémantique par patchs, nous avons dû adapter la base d'apprentissage et prélever sur chacune des images des patchs libellés « ticket » et « non ticket ». Aucun patch n'est prélevé à la frontière des deux classes.

Cependant le choix de la taille des patchs ainsi extraits est très important, car chaque patch doit contenir suffisamment d'informations discriminantes et représentatives d'un ticket de caisse. Or, rappelons-le, les images de la base de données proviennent de terminaux mobiles dont la résolution moyenne est de 8 Mpx. Les images sont donc relativement grandes et si l'on choisit directement des patchs de taille « 227 x 227 », taille des images de la base d'apprentissage du réseau AlexNet [31], ces patchs peuvent ne contenir aucune information (patch blanc) à cause des larges espaces entre certains blocs de texte, comme illustré sur la figure 4.9. Afin de déterminer la taille de patch permettant d'assurer que

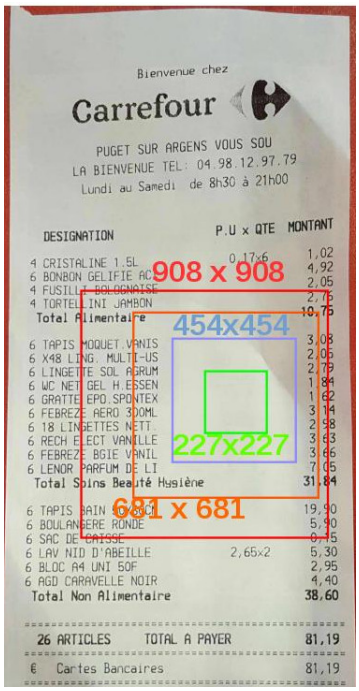


FIGURE 4.9. – Évaluation de la taille des images à extraire permettant d'assurer la présence de suffisamment d'informations.

tous les patchs extraits contiennent suffisamment d'informations discriminantes, nous avons expérimenté, sur des images de tickets de grande résolution (car c'est pour ces images que le problème est le plus critique) plusieurs tailles de patchs : « 227 x 227 », « 454 x 454 », « 681 x 681 »,« 908 x 908 ». Le meilleur compromis a été la taille « 681 x 681 ». Choisir un patch trop grand risque d'être gênant sur les images de plus petites résolutions. Ces patchs sont ensuite redimensionnés à la taille « 227 x 227 » pour correspondre à la taille d'entrée des CNN utilisés.

De cette manière, dans chaque image de la base d'apprentissage , dix imagettes de taille carrées ont été extraites au hasard dans la zone « ticket » et dix imagettes dans la zone « non-ticket » de l'image.

Ainsi la base d'apprentissage contient 35 000 imagettes. L'apprentissage a été réalisé sur 23 400 images de la base annotée, les 11 600 images restantes servant à la phase de test. Nous avons effectué une validation croisée (trois apprentissages en renouvelant la base par 1/3). La performance du réseau est alors mesurée par la moyenne des performances obtenues après chacun des trois apprentissages. La performance d'un modèle est le pourcentage de bonne classification sur l'ensemble de tests.

L'entraînement des quatre CNN s'est effectué à l'aide du framework Caffe.

Le « learning rate » (« base_lr ») est initialisé à 10^{-3}. Nous avons défini le « step-

Architecture	Configuration	Accuracy
AlexNet	**AlexNet_lastFC**	0.9693 ± 0.0083
	AlexNet_3FC	0.9715 ± 0.0074
GoogLeNet	**GoogLeNet_3Classifer**	**0.9887 ± 0.0052**
	GoogLeNet_LastClassifer	0.9791 ± 0.0055

TABLE 4.2. – Précision de classification pour la reconnaissance de ticket dans une image pour les quatre architectures testées.

size » à 2000 c'est-à-dire que toutes les 2000 itérations celui-ci est divisé par 10 (valeur de « gamma »).

Partant d'un « learning rate », la modification du « learning rate » au cours de l'apprentissage permet d'affiner la convergence du modèle quand le gradient atteint un minimum local. L'entraînement est arrêté au bout de 10 000 itérations (*max_iter*), valeur obtenue empiriquement.

Choix du modèle : Compte-tenu des résultats obtenus, les deux réseaux basés sur l'architecture GoogLeNet permettent d'atteindre les meilleures précisions. Nous avons donc fait le choix d'utiliser le réseau GoogLeNet « **GoogLeNet_LastClassifer** », car son entraînement est réalisé à moindre coût, avec une unique couche à ré-entraîner. Le réseau a donc été transformé en un réseau complètement convolutionnel en suivant la méthode décrite au paragraphe 4.2.2.

Méthode d'analyse : Les images envoyées au système d'analyse de tickets proviennent de terminaux mobiles divers et variés et dont la résolution des images varie fortement. Afin d'avoir une certaine homogénéité sur les temps de traitement, il était important de redimensionner les images avant de les analyser.

Nous avons testé plusieurs tailles différentes illustrées sur la figure 4.11, avant de s'arrêter sur la taille de « 1219 x 1219 » pixels les raisons suivantes :

— Sur cette figure 4.11, nous pouvons remarquer que plus la taille de la carte de chaleur est grande, plus la segmentation est précise. Cependant, pour les tailles importantes bien que la réponse soit plus précise, le temps de calcul est trop important (jusqu'à 50ms). La taille choisie correspond au meilleur compromis entre temps de traitement et précision de la réponse ;

— La dimension d'image choisie en entrée a également pour but de diminuer, encore une fois, le nombre de fenêtres contenant peu d'informations discriminantes à l'intérieur du ticket (zones blanches). En effet, en réduisant la dimension de l'image, les risques d'obtenir des patchs ne contenant que le fond blanc du ticket (sans texte) sont minimisés ;

— Nous avons expérimentalement constaté que la taille carrée déforme certes l'apparence du ticket (voir figure 4.10), mais que cela avait très peu d'impact sur la classification des différents patchs. Le choix d'une taille rectangulaire était plus

FIGURE 4.10. – Visualisation de la déformation légère du ticket dans l'image lorsque celle-ci est redimensionnée en une taille carrée de « 1219 x 1219 ».

risqué, car a priori nous n'avons aucune information sur le sens du ticket dans l'image.

Comme le réseau choisi, « GoogLeNet », possède un champ de vision de 32 pixels, la carte de chaleur obtenue en sortie du réseau de segmentation est de dimension 32 x 32 pixels. Afin de calculer la dimension de la carte de chaleur en fonction de la taille de l'image en entrée, il suffit d'appliquer l'équation suivante :

$$TailleCarteChaleur = \frac{TailleImageEntree - 227 + ChampVision}{ChampVision} \tag{4.1}$$

où « 227 », correspond à la taille de la fenêtre coulissante où encore la taille de l'image d'entrée du réseau de classification.

Une fois que le réseau a traité une image, la carte de chaleur résultante consiste en une image en niveau de gris dont les valeurs, comprises entre 0 et 1, représentent la probabilité qu'a un pixel d'appartenir à la classe « ticket ». Cette image en niveau de gris est binarisée en prenant un seuil égal à 0.5 tout naturellement.

Ensuite, le pourcentage de pixels positifs « $P(pixels \in ticket)$ » , autrement dit « blancs » est calculé. Nous avons tracé, sur le graphique de la figure 4.12, les valeurs de ce pourcentage de pixels positifs (axe des ordonnées) sur un échantillon de 1 500 images correspondant à des images avec ticket et des images sans ticket. Rappelons que notre objectif ici est de minimiser la perte d'images contenant un ticket et de purifier au maximum le système d'analyse de tickets en éliminant les images non analysables (sans ticket). Pour répondre à cet objectif, nous avons défini deux seuils, un « seuil supérieur » en vert et un « seuil inférieur » en violet sur la figure 4.12 :

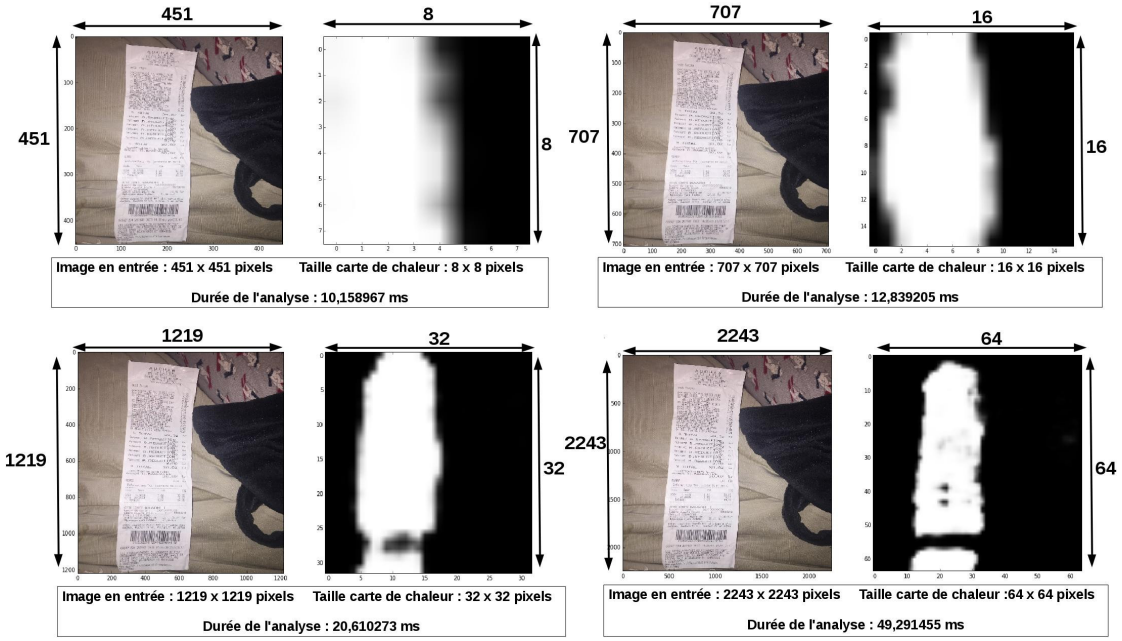


FIGURE 4.11. – Évaluation des différentes tailles de carte de chaleur. Comparaison entre la précision de la réponse et le temps de calcul.

— $P_{superieur} = P(pixels \in ticket) > 25\% =>$ Si le pourcentage de pixels positifs est supérieur à 25%, alors l'image contient un ticket ;
— $P_{inferieur} = P(pixels \in ticket) < 5\% =>$ Si le pourcentage de pixels positifs est inférieur à 5%, alors l'image ne contient pas de ticket.

Le seuil de $P(pixels \in ticket) = 25\%$ peut paraître faible à première vue, cependant sur les images contenant de très longs tickets, la surface occupée par le ticket est relativement faible comme illustré sur la figure 4.13 où le pourcentage est tout juste à 26%. À titre de comparaison, des expériences (comme nous pouvons le constater sur le graphique illustré sur la figure 4.12) ont montré que le taux de pixels positifs des images « sans ticket » dépasse rarement les 5%. Les images dont le pourcentage $P(pixels \in ticket) < 5\%$ sont donc rejetées.

Cependant les images dont le pourcentage vérifie les conditions suivantes : $P(pixels \in ticket) < 25\%$ et $P(pixels \in ticket) >= 5\%$, ne sont pas rejetées immédiatement. Encore une fois, l'objectif est de ne pas manquer de ticket. À partir de la carte de chaleur, nous sommes capables de localiser grossièrement la zone qui correspondrait à un ticket de caisse. Cette zone de l'image est alors soumise au réseau de classification, afin de vérifier si celle-ci contient un ticket ou non. Cette double vérification nous permet de ne pas manquer les tickets comme illustrés sur la figure 4.14.

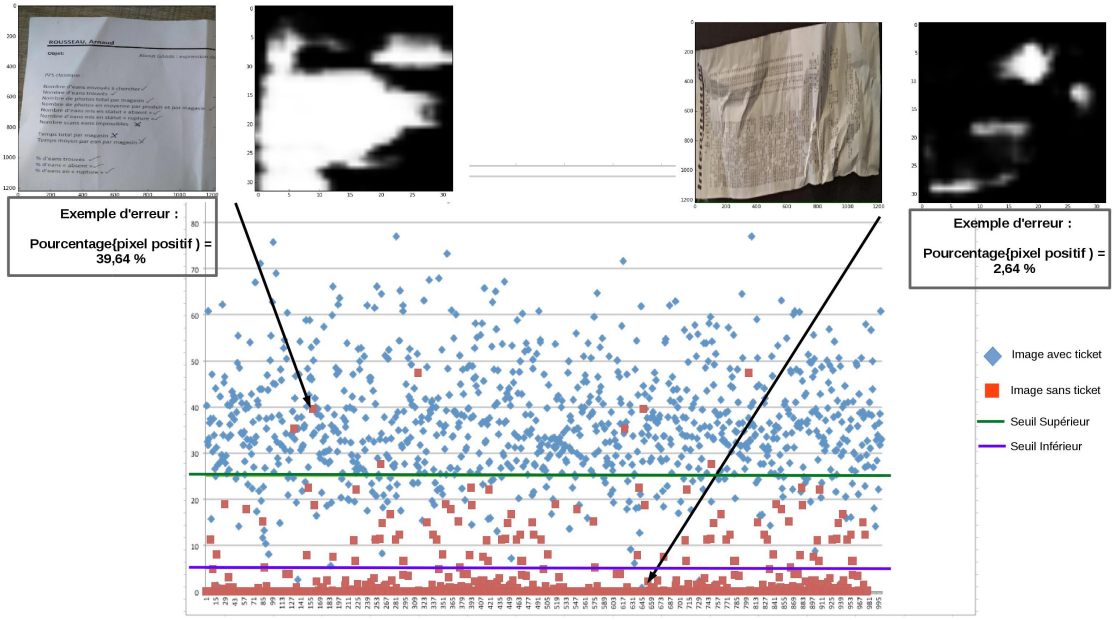


FIGURE 4.12. – Graphique représentant le pourcentage de pixels positifs mesuré sur la carte de chaleur, lors de l'étape « détection du ticket dans l'image » sur un échantillon de 1 500 images. En bleu est représenté le pourcentage calculé sur des images contenant un ticket, en rouge sur des images ne contenant pas de ticket. En vert est représenté le « seuil supérieur » et en violet le « seuil inférieur » défini expérimentalement.

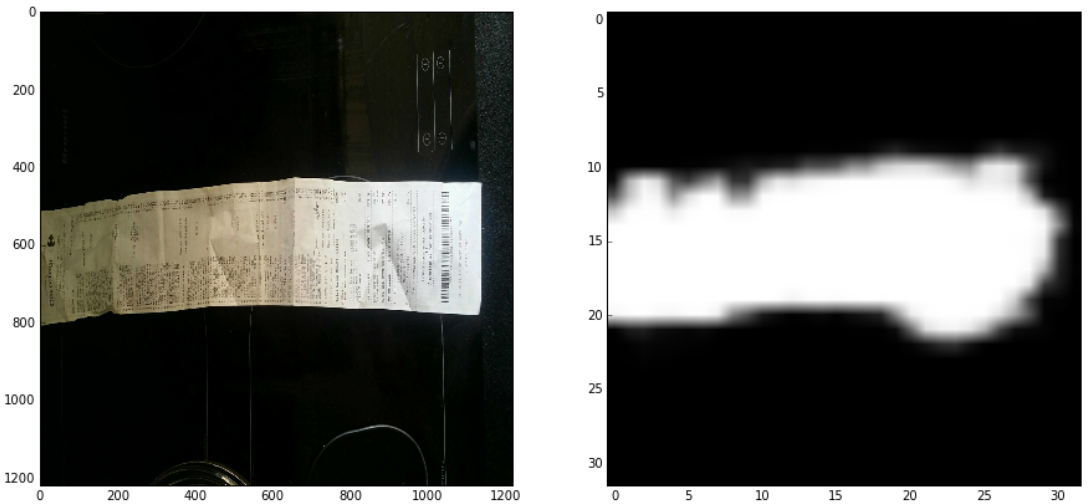


FIGURE 4.13. – Sur ce ticket de longueur moyenne, le pourcentage de pixels positifs (pixel appartenant au ticket) est seulement de 26%.

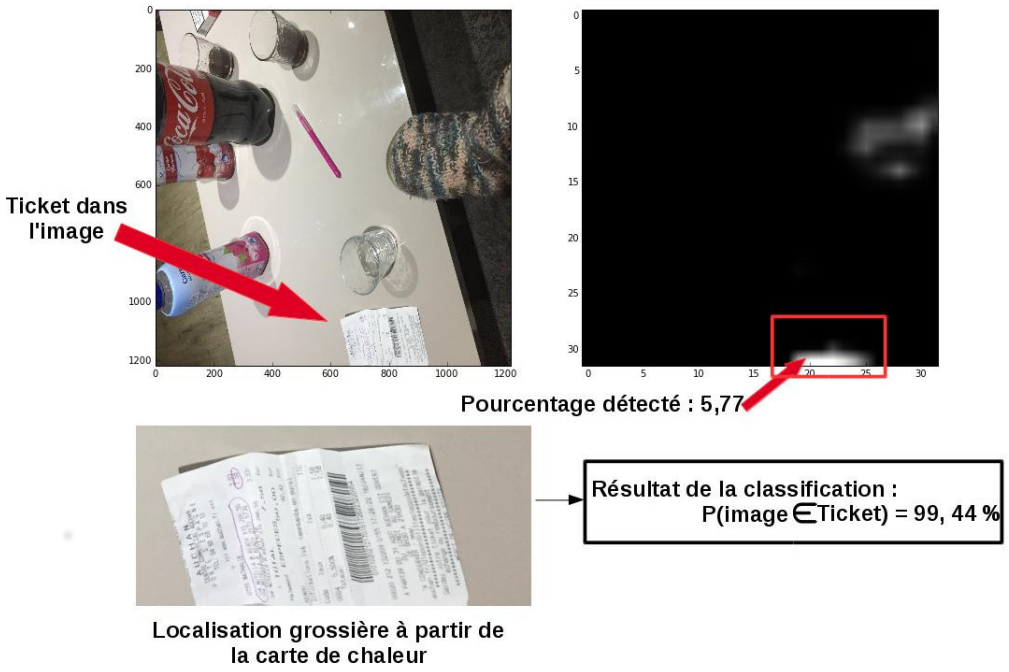


FIGURE 4.14. – Détection d'un ticket en deux temps, malgré que celui-ci soit enfoui dans l'image.

Synthèse

Sur le tableau 4.3, nous avons reporté les pourcentages de bonne classification et la matrice de confusion pour la méthode de segmentation sémantique présentée ci-dessus et la méthode, plus simple et directe, utilisant le réseau de neurones en mode « classification ». Pour les deux méthodes, le même réseau de neurones est mise en jeu, celui que nous avons entraîné : « GoogLeNet_3Classifier » (cf tableau 4.2). Pour la méthode de classification, l'image brute en entrée du réseau de neurones est redimensionnée à la taille « 227 x 227 » pixels avant d'être classifiée, tandis que la méthode de segmentation sémantique suit le protocole décrit au paragraphe ci-dessus 4.3.1.

Nous constatons que la méthode basée sur la segmentation sémantique permet d'obtenir de meilleurs résultats. Comme nous l'avons déjà dit, les images à classifier (voir figure 4.2) contiennent souvent énormément d'informations et le ticket est enfoui dans l'image ce qui complexifie la tâche. Ces résultats justifient le choix d'utiliser un réseau de segmentation sémantique afin de valider la présence d'un ticket dans l'image. De plus que l'application de cette méthode sera très utile pour la prochaine étape qui consiste à localiser le ticket dans l'image.

	Classification directe par réseau de neurones		Classification par une segmentation sémantique par patchs		
Accuracy	95.87%		99,00%		
Matrice Confusion	**Ticket**	**NonTicket**	**Ticket**	**NonTicket**	**Tot.**
Ticket	985	15	996	4	1000
NonTicket	47	453	1	489	500

TABLE 4.3. – Précision de la classification et matrice de confusion pour la détection d'un ticket sur une image, selon deux méthodes différentes : méthode de classification directe par un réseau de neurones et une la méthode que nous avons proposée, basée sur de la segmentation sémantique par patch. Résultats obtenus sur une base de données de 1500 images.

4.3.2. Méthode proposée pour l'analyse de texte

Une autre méthode permettant la détection d'un ticket de caisse dans une image est l'analyse de texte en utilisant un outil d'OCR (Reconnaissance Optique de Caractères) appliqué sur l'ensemble de l'image. Les expérimentations que nous avons menées lors de la construction du démonstrateur développée au chapitre 2, au paragraphe 2.3.4 ont permis de mettre en évidence que l'OCR le plus intéressant pour notre problème est celui proposé par Google à savoir GoogleVision [21]. Malgré son aspect boîte noire payante, nous avons fait le choix d'utiliser cet OCR (adapté pour l'usage en entreprise). Bien que l'OCR soit performant, à ce niveau du processus, les performances ne sont pas suffisantes pour pouvoir effectuer une analyse fine du contenu du ticket à cause des perturbations apportées par l'arrière-plan, comme des motifs qui apportent de fausses détections, mais également la mauvaise orientation du ticket, comme nous pouvons le voir sur la figure 4.15. Cependant, l'usage des critères simples sur une analyse OCR de faible qualité est suffisant pour valider la présence d'un ticket dans l'image.

D'un point de vue sémantique, une image contient un ticket si le texte fourni par l'OCR contient au moins une « ligne de produits ». Une ligne de produits est une séquence de caractères correspondant à un format de chaîne très spécifique : typiquement un ensemble de lettres, chiffres, espaces et ponctuations sur la première partie, suivi d'espaces et finissant par une chaîne respectant un format de prix (chiffres éventuellement avec une virgule ou un point et éventuellement un symbole monétaire) Exemple :
« BRICK LP 0,79 € »
Cette détection est effectuée en utilisant des expressions régulières afin de couvrir les différents formats de « lignes produits » dans les tickets de caisse. Il suffit qu'une unique ligne de produits soit détectée pour affirmer la détection d'un ticket.

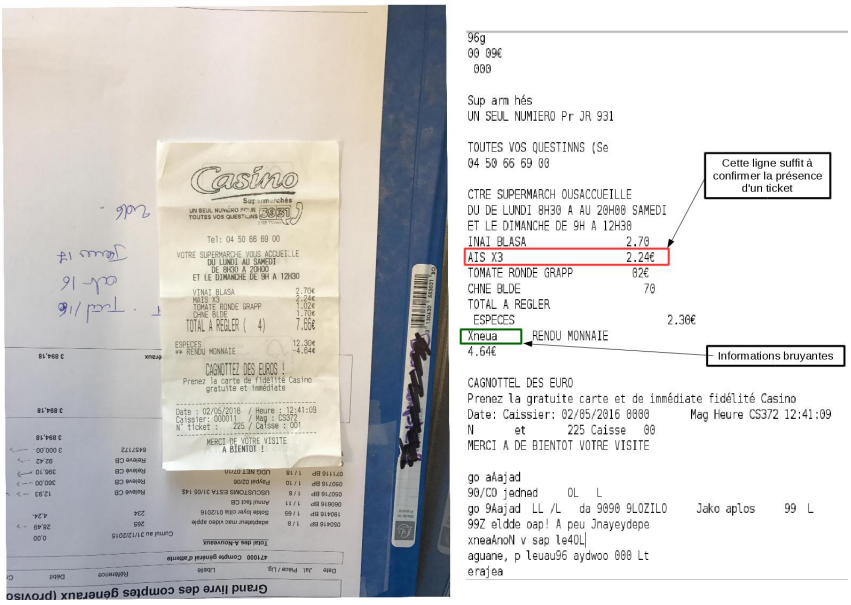


FIGURE 4.15. – Résultat de la sortie de l'OCR « Google Vision » sur une image avec un fond bruité. La détection d'une « ligne produit » est possible, mais l'analyse du contenu est fortement compromise.

Étant donné que nous ne disposons d'aucune information a priori sur l'acquisition de l'image, il est possible que l'opération soit répétée au plus quatre fois, afin d'évaluer l'image dans toutes les orientations possibles.

Si le texte contenu sur le ticket est correctement lu, la détection de la présence d'une « ligne produit » est toujours garantie. Les tests effectués sur 1 200 tickets parfaitement lus ont donnée 100% de bonnes détections des « lignes produits ». En revanche les performances se dégradent si la lecture faite par l'OCR est imparfaite.

4.3.3. Fusion des deux méthodes

Les deux algorithmes permettant de détecter la présence d'un ticket présentés ci-dessus ont pour avantage d'être complémentaires. En effet lorsque nous analysons de plus prêts les cas d'erreurs types pour chacun des deux algorithmes, les situations sont différentes, comme illustré sur la figure 4.17. L'analyse d'image est généralement trompée par des images contenant un document ou une partie de document ayant une typographie ressemblante à celle d'un ticket de caisse (du texte noir sur un fond blanc). L'analyse de texte, elle, est très dépendante de la sortie de l'OCR. Si le ticket est enfoui dans l'image, il n'est pas rare que la sortie de l'OCR soit nulle. Dans ce cas-là, l'analyse de texte est incapable d'affirmer ou non la présence d'un ticket dans l'image.

Afin de fusionner les deux résultats, plusieurs choix sont possibles : l'utilisation des opérateurs « OU logique », « ET logique », ou encore définir une loi en fonction du taux

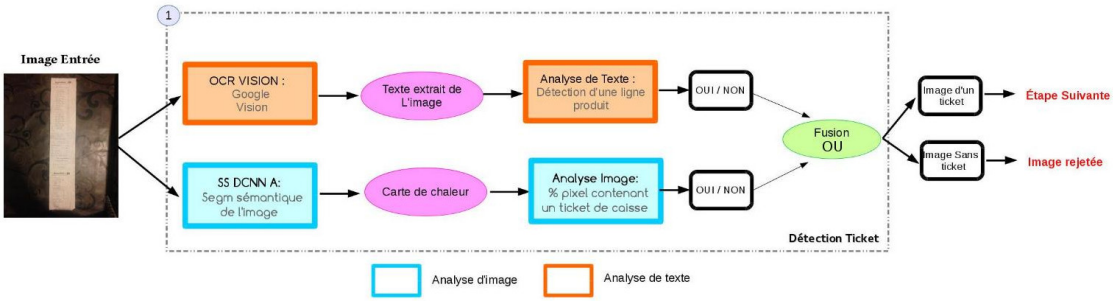


FIGURE 4.16. – Description de l'approche proposée pour la détection d'un ticket dans l'image, fusion de deux analyses différentes.

de confiance de chaque algorithme. Prenant en compte la contrainte industrielle de ne manquer aucun ticket, nous avons naturellement fait le choix d'utiliser l'opérateur « OU logique ». En effet l'opérateur « ET logique » est trop stricte, allant à l'encontre de la contrainte, et la définition d'une loi en fonction du taux de confiance demanderait de disposer d'un taux de confiance de chaque algorithme.
La solution « OU logique » permet de minimiser encore une fois le taux d'images de ticket rejetées tout en améliorant la confiance de détection. Le seul risque est que le système analyse certaines images ne contenant pas de ticket. Mais l'image en question sera rejetée dans la suite du traitement qui cherche des informations spécifiques (enseigne, date d'achat, montant total, listes des produits ...) qui ne seront pas trouvées.
Bien évidement, un taux de confiance peut être accordé à la réponse de cette étape. Si les deux algorithmes sont d'accord, le taux de confiance est élevé. Sur la base de test, le pourcentage de bonne classification lorsque les deux algorithmes sont d'accord est de 100% sur 5 000 images acquises par des utilisateurs réels de l'application.

4.3.4. Résultats et Performances

Nous avons présenté une méthode ayant pour objectif de détecter la présence d'un ticket dans une image en fusionnant le résultat de deux algorithmes réalisés en parallèle. La description résumée de l'approche proposée est illustrée sur la figure 4.16, où le réseau de segmentation sémantique est nommée « SS DCNN A ».

Les performances des deux algorithmes décrits pris indépendamment ainsi que la performance obtenue après fusion sont reportées dans le tableau 4.4, en donnant les mesures de « précision » et de « rappel » pour chacune des classes.

Nous avons obtenu ces performances sur une base de 5 000 images, manuellement annotées, contenant 3 000 images avec un ticket de caisse et 2 000 images sans ticket de caisse.

Le contexte industriel nous interdisant la perte d'image contenant un ticket de caisse, nous mettons un accent particulier sur le « rappel » que nous souhaitons le plus élevé possible.

Précision			
Classe	**DCNN**	**Text Sem.**	**DCNN & Text Sem.**
Ticket	98.5%	98.7%	99.6%
Non Ticket	90%	67%	93%
Rappel			
Classe	**DCNN**	**Text Sem.**	**DCNN & Text Sem.**
Ticket	99.5%	92.8%	99.9%
Non Ticket	91.3%	91.07%	96.8%

TABLE 4.4. – Performance de la détection du Ticket.

La fusion entre l'approche « analyse de texte » et l'approche « analyse d'image » améliore les performances globales, car les cas d'erreur sont différents d'une approche à l'autre. L'analyse sémantique du texte peut ne pas détecter un ticket si l'image est légèrement floue ou si l'OCR ne reconnaît pas le texte, alors que les CNNs mis en œuvre dans l'analyse d'image peuvent détecter des tickets sur des documents ayant la même typographie que les tickets de caisse. Les différents cas d'erreur rencontrés sont illustrés sur la figure 4.17.

L'étape suivante est la localisation du ticket dans l'image.

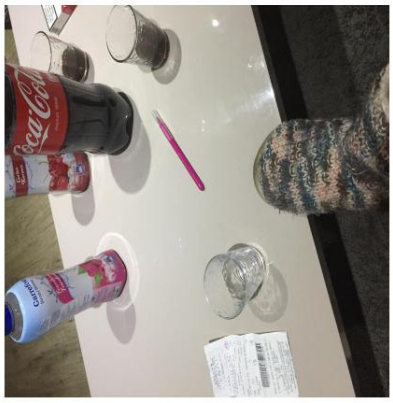
4.4. Localisation du ticket

4.4.1. Méthode proposée pour la localisation du ticket

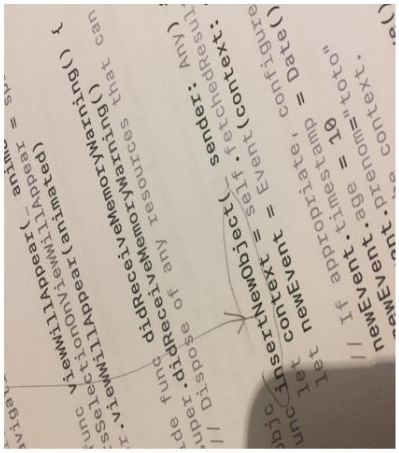
Une fois que toutes les images « sans ticket » ont été rejetées, la deuxième étape est la localisation du ticket dans l'image, afin d'obtenir une image rognée, débarrassée de son arrière-plan. Cette étape est très liée à la première puisque sa réalisation ne nécessite pas l'utilisation d'un nouveau modèle. En effet, l'exploitation de la carte de chaleur fournie par le réseau de neurones de l'étape précédente, nommé « SS DCNN A » est suffisante.

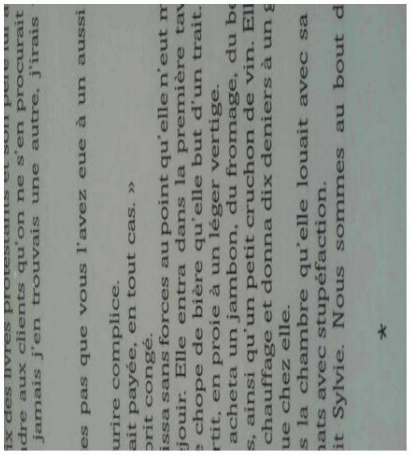
Cependant plusieurs étapes de calcul sont nécessaires afin d'obtenir la localisation précise du ticket dans l'image. Ces étapes sont coûteuses en temps de calcul, c'est pourquoi il est utile de réaliser cette étape de localisation une seule fois grâce à la carte de chaleur obtenue à l'étape précédente.

Après une localisation grossière du ticket de caisse dans l'image, l'objectif est d'affiner cette localisation afin de délimiter précisément les contours du ticket.



Cas d'erreur de l'analyse sémantique : Aucun texte n'est détecté par l'OCR : ticket trop petit et mal orienté pour permettre l'analyse.





Cas d'erreur du CNN : Texte noir sur fond blanc/typographie ressemblant à un ticket

FIGURE 4.17. – Les cas d'erreurs de l'approche « analyse d'image » et de l'approche « analyse de texte » concernant la détection de la présence d'un ticket dans l'image.

4.4.1.a. Localisation grossière par DL

La carte de chaleur obtenue à la première étape de détection de ticket, grâce au réseau de segmentation sémantique « SS DCNN A », nous permet de localiser de manière approximative le ticket de caisse et d'éliminer une large partie du fond de l'image (voir figure 4.18) .

Le seuil « $P_{seuil} = P(ticket \in image) > 50\%$ » est réutilisé, afin de définir la boîte englobant tous les pixels de la carte de chaleur répondant à cette condition.

Compte tenu de la résolution de l'image en entrée : « 1219 x 1219 » et de la taille des fenêtres coulissantes : « 227 x 227 » , la boîte englobante ainsi définie est toujours plus large que le ticket. De cette manière, nous pouvons assurer qu'il n'y a aucune perte d'informations, comme illustrée sur la figure 4.18 .

Cette étape permet de réduire considérablement le contexte et d'éliminer un maximum d'informations inutiles.

4.4.1.b. Localisation fine des contours

Afin d'améliorer la localisation et obtenir des coordonnées plus précises des contours du ticket, nous avons exploré plusieurs méthodes.

1ère méthode : La première méthode que nous avons proposée est de répéter le processus de segmentation sémantique sur l'image en sortie, afin d'affiner la localisation. Autrement dit, l'idée est de gagner en précision, en segmentant autant de fois que nécessaire l'image initiale. La condition d'arrêt est tout simplement l'égalité des tailles entre les images d'entrée et de sortie du réseau. Cette méthode est nommée « bouclage DL statique ».

Comme nous pouvons le voir sur la figure 4.19, le seul changement à chaque tour de boucle est l'image nommée « Image initiale » à l'entrée du réseau de segmentation qui se recentre de plus en plus sur le ticket.

2ème méthode : La seconde méthode nommée « bouclage DL progressif » est très proche de la première dans le sens où l'idée de bouclage sur le même réseau de segmentation sémantique est conservée. Cependant, à chaque itération la résolution de l'image en entrée, et donc de la carte de chaleur (cf equation 4.1) est augmentée, pour que la segmentation soit de plus en plus fine à chaque étape. Comme pour la première méthode, le processus est arrêté lorsque les tailles de l'image en entrée et en sortie sont identiques. Cependant, nous avons fixé une taille d'image d'entrée maximale à « 2243 x 2243 » car, au-delà, la déformation de l'image serait trop importante. Cette dimension maximale, nous fixe le nombre de re-bouclage maximum à 5, puisque nous augmentons la taille de la carte de chaleur par pas de 8 en partant de la taille initiale de « 32 x 32 » pixels. Dans le tableau 4.5, nous reportons les différentes dimensions, à chaque tour de boucle.

Comme nous pouvons le voir sur la figure 4.20, le réseau de segmentation sémantique mis en jeu est exactement le même que celui de la solution précédente. Il a la particularité

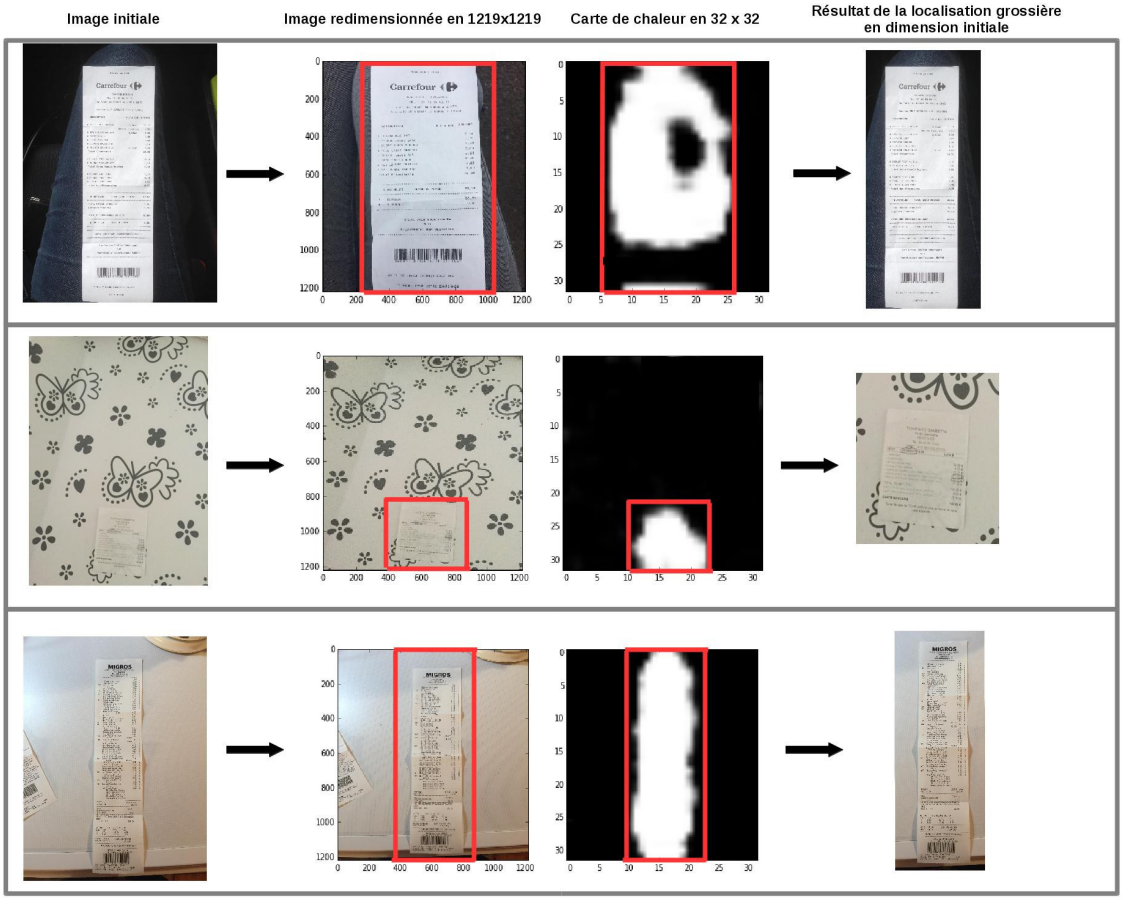


FIGURE 4.18. – Quelques résultats de la localisation grossière des tickets de caisse grâce à l'analyse de la carte de chaleur obtenue en sortie du réseau de segmentation sémantique. On ne constate aucune perte d'informations, mais une localisation imprécise.

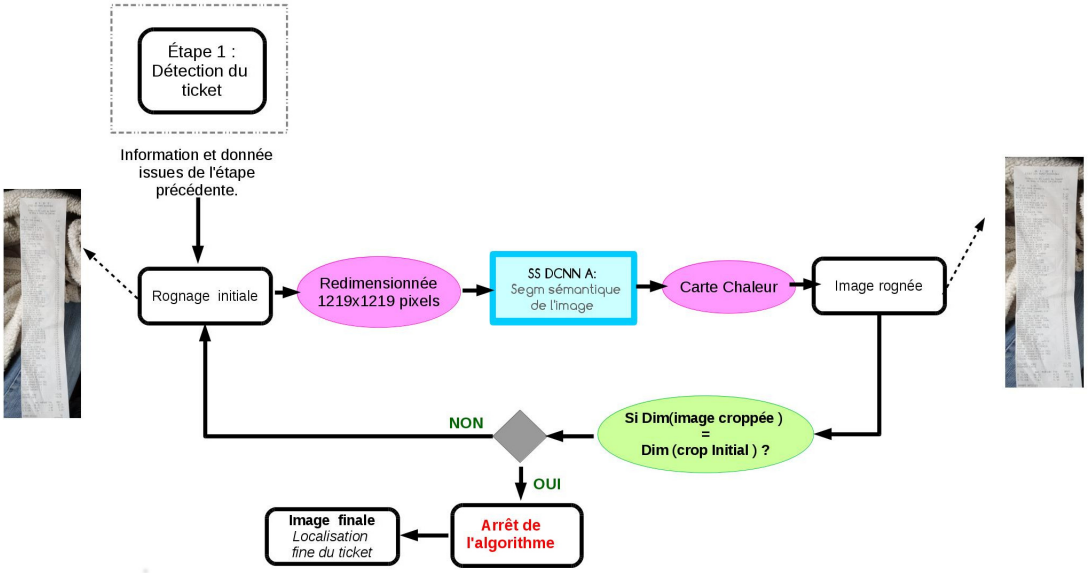


FIGURE 4.19. – Algorithme de localisation fine du ticket, nommé « bouclage DL statique »

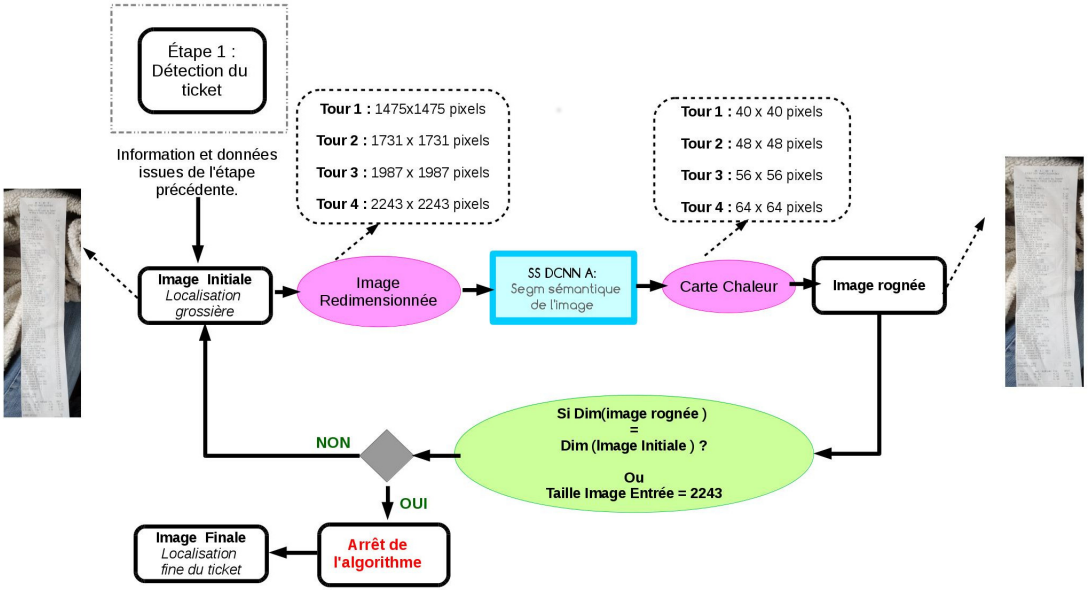


FIGURE 4.20. – Algorithme de localisation fine du ticket, nommé « bouclage DL progressif »

Num. Boucle	1	2	3	4	5
Taille image entrée	1219x1219	1475x1475	1731x1731	1987x1987	2243x2243
Taille carte chaleur	32x32	40x40	48x48	56x56	64x64

TABLE 4.5. – Différentes dimensions évaluées pour la localisation du ticket dans l'image pour la méthode « bouclage DL progressif »

de pouvoir prendre en entrée n'importe quelle dimension d'image, le seul changement est la dimension de la carte de chaleur en sortie de ce réseau qui, elle, est directement liée à la taille de l'image en entrée (cf équation 4.1). Attention, le réseau de segmentation n'est pas pour autant un réseau multi-échelle, puisqu'il classifie uniquement des patchs d'images de taille « 227 x 227 ». Seule l'échelle de l'image d'entrée change.

Cependant plus la taille de l'image en entrée est grande, plus le temps de calcul est long, comme nous avons pu le constater lors des expérimentations illustrées sur la figure 4.11.

3ème méthode : La troisième méthode consiste à n'utiliser le réseau de segmentation qu'une seule fois, celle de l'étape précédente (détection de ticket dans l'image). Ensuite, afin d'affiner la localisation, nous utilisons le détecteur de contour que nous avons développé spécifiquement [132]. Les caractéristiques de ce détecteur ont été présentées au chapitre 2 à la section 2.2.4.b. Pour rappel, le principe de base de ce détecteur des bords d'un ticket est de rechercher des régions avec deux zones adjacentes principales, une zone claire (la bordure du ticket) et une plus sombre (l'arrière-plan), en position quasi verticale ou horizontale. Une fois les 4 arêtes obtenues, des considérations géométriques simples (recherche d'une forme rectangulaire) permettent de définir le contour précis du ticket de caisse. Tous les paramètres mis en jeu sont développés au chapitre 2 section 2.2.4.b). Les différentes étapes et le résultat de cette méthode sont illustrés sur la figure 4.21. Néanmoins cette dernière méthode demande d'avoir des contours du ticket peu décalés par rapport à la verticale ou l'horizontale.

4.4.2. Résultats et Performances

Pour évaluer les différentes méthodes proposées, nous avons utilisé une base d'images construite à partir d'images recueillies par l'entreprise (cf 4.2.3) et sur lesquelles nous avons manuellement annoté la position du ticket dans l'image, sous forme de coordonnées d'un rectangle. Cette base contient environ 1 100 images.

Les performances sont évaluées en utilisant la métrique « Intersection Over Union ». Dans le tableau 4.6 sont reportés les performances et temps de calculs obtenus pour chacune des méthodes proposées (les tests ont été réalisés sur une machine avec GPU 8Go).

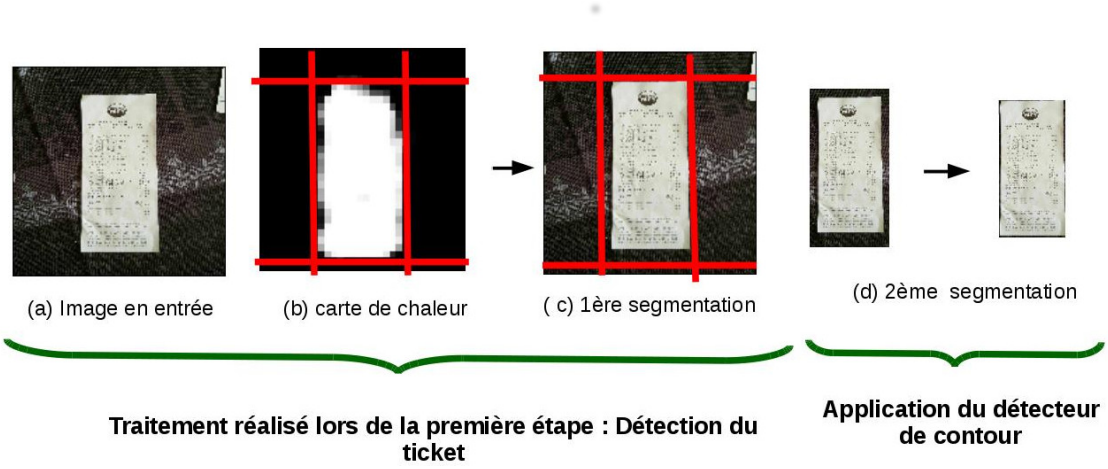


FIGURE 4.21. – Localisation du ticket en deux étapes via la méthode « Deep + Détecteur ».

Localisation du ticket et rognage - IoU		
Méthode	**IoU Moyen**	**Temps de calcul**
Deep Uniquement	0.68	20 ms
Détecteur Uniquement	0.72	3ms
bouclage DL statique	0.83	125 ms
bouclage DL progressif	0.91	179ms
Détecteur & Deep	**0.87**	**25ms**

TABLE 4.6. – Performances de la localisation des tickets de caisse.

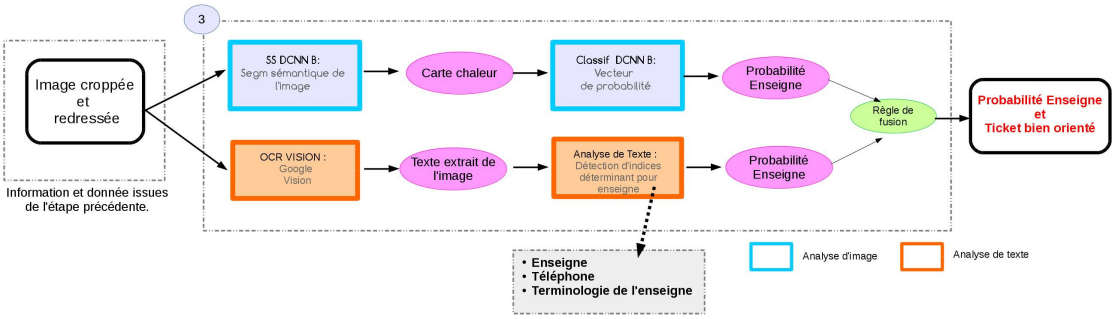


FIGURE 4.22. – Détection de l'enseigne, fusion de deux analyses différentes.

Le meilleur compromis entre niveau de performance et temps de calcul est obtenu par la méthode qui combine l'utilisation de la carte de chaleur du Deep puis le détecteur de contours. En effet, la combinaison des deux méthodes permet d'obtenir une importante amélioration des performances de chacune des méthodes prises indépendamment, sans augmenter pour autant le temps de calcul. L'intérêt de la carte de chaleur obtenue par le réseau de segmentation sémantique est démontré lorsque l'on compare la moyenne IoU obtenue par le détecteur seul (0.72) et celle obtenue pour la méthode combinant « Deep & Détecteur » (0.87). La carte de chaleur obtenue via l'application du réseau de neurones élimine une large partie de l'arrière-plan de l'image et réduit considérablement le nombre de fausses détections du détecteur de contours.
On peut également noter de meilleures performances performance atteinte avec la méthode « bouclage DL progressif » : IoU = 0,91 mais au prix d'un temps de calcul important.

4.5. Détection et reconnaissance du logo

La troisième et dernière étape du pré-traitement est l'identification de l'enseigne du ticket de caisse. Comme pour la première étape, cette étape met en jeu, à nouveau, deux algorithmes en parallèle dans le but de renforcer la fiabilité du système. La première approche correspond à une analyse d'image et consiste à détecter le logo dans l'image, tandis que la deuxième approche concerne l'analyse du contenu textuel afin de détecter différents indices permettant de déterminer l'enseigne. Ces deux étapes sont complémentaires car sur certains tickets le logo est remplacé par la simple écriture du nom de l'enseigne. Un aperçu de l'approche proposée est illustré sur la figure 4.22.

4.5.1. Méthode proposée

4.5.1.a. Analyse d'image

À cette étape, l'analyse d'image a deux objectifs. Elle doit dans un premier temps localiser le logo dans le ticket et ensuite le catégoriser afin de déterminer l'enseigne.
Nous avons vu au chapitre 3 (paragraphe 3.3), parmi les différentes approches présentées dans la littérature que les méthodes basées sur le Deep Learning sont les plus appropriées pour ce type de problématique.

L'approche que nous proposons repose donc essentiellement sur un réseau de neurones profond de classification « Classif DCNN B » et de sa transformation en réseau 100% convolutionnel « SS DCNN B » permettant d'obtenir une segmentation sémantique de l'image par patchs afin de localiser le logo dans l'image.

Choix du réseau de classification : comme pour l'étape de détection de tickets, nous avons ici comparé les performances des quatre architectures de CNN présentées à la section 4.2.1.c.

Concernant la base d'apprentissage, nous disposons pour ce problème de classification d'une base réduite représentant 17 classes de logo, avec une centaine d'exemples par logo (voir section 4.2.3). Comme on peut le voir sur la figure 4.23, les conditions d'éclairage et les terminaux mobiles utilisés étant différents d'une image à l'autre, la luminosité, la qualité et même la couleur de fond varient. Comme les logos recherchés sont des motifs sombres sur fond clair sans information de couleur et afin que cette variété de conditions d'acquisition n'influe pas trop sur la classification, nous avons converti toutes les images en niveau de gris. Cette manipulation simple est d'autant plus justifiée que la base d'apprentissage est peu importante. En homogénéisant les différents exemples d'une classe, on ne peut que faciliter l'apprentissage.

Pour la répartition des données entre la phase d'apprentissage et la phase de test, nous avons respecté la proportion suivante : 2/3 des données pour l'apprentissage et 1/3 des données pour le test.

L'entraînement des quatre CNNs a été fait avec le framework Caffe et nous avons utilisé le même principe de « cross validation » que pour la détection du ticket dans l'image. Les résultats obtenus sont présentés dans la table 4.7. Dans ce tableau, nous avons reporté le pourcentage de bonne classification en « Top-1 » et en « Top-5 ».

Compte tenu des résultats obtenus, nous avons fait le choix d'utiliser le réseau GoogLeNet « **GoogLeNet_LastClassifier** ». Même si les performances du réseau « AlexNet_3FC » sont très proches du réseau sélectionné, le réseau GoogLeNet met en jeu beaucoup moins de paramètres, il sera donc plus léger à embarquer sur un appareil mobile (objectif en perspective).

FIGURE 4.23. – Échantillon des logos de la base d'apprentissage.

Architecture	Configuration	Prec. Top-1	Prec. Top-5
AlexNet	**AlexNet_lastFC**	0.9210±0.0090	0.9863±0.0066
	AlexNet_3FC	0.9763±0.0119	0.9986±0.0010
GoogLeNet	**GoogLeNet_3Classifer**	0.9100±0.0141	0.9815±0.0096
	GoogLeNet_LastClassifer	**0.9798±0.0054**	**0.9933±0.0041**

TABLE 4.7. – Précision de la classification pour la reconnaissance des logos dans une image de ticket de caisse pour les quatre architectures testées.

Méthode d'analyse : En entrée de cette étape les images de ticket de caisse à analyser sont rognées et redressées.

L'approche que nous proposons pour la détection du logo se fait en plusieurs étapes.

Étape 1 : Pré-traitement de l'image

Parmi tous les logos des enseignes considérées nous avons pu observer (voir illustration sur la figure 4.23), qu'il existe deux types de logos : les longs et les courts, tous étant de forme rectangulaire avec les côtés les plus longs sur l'axe horizontal. Une analyse des dimensions précises des différents logos nous a permis de définir deux rapports typiques « largeur / hauteur » ($rapport_1 = \frac{8}{5}$ et $rapport_2 = \frac{18}{5}$) représentant bien l'ensemble de ces deux situations et de redimensionner les images afin de générer au moins une version d'un logo carré qui s'intégrera dans une image de 227 x 227 pixels pour alimenter le réseau profond de segmentation sémantique. La figure 4.25 montre une illustration d'un redimensionnement approprié à l'étape 1.
Il faut également mentionner que les deux facteurs de redimensionnement sont choisis afin de s'assurer que la fenêtre glissante générée par le « SS DCNN B » contienne, au moins une fois, le logo complet. Plus précisément, nous rappelons que lors de l'application du réseau de segmentation sémantique « SS DCNN B », une fenêtre coulissante de taille « 227 x 227 » parcours l'image avec un pas de « 32 x 32 » correspondant au champ récepteur de l'architecture. Sans redimensionnement de l'image en amont, les chances qu'une des fenêtres coulissantes contiennent suffisamment d'informations pour identifier le logo sont très pauvres. D'un point de vue pratique, cela permet aussi de gagner du temps (les images à traiter étant plus petites) . Comme les logos longs sont plus fréquents, nous utilisons d'abord le rapport de redimensionnement correspondant à ce type de logos. Si le réseau de segmentation sémantique (« SS DCNN B », étape 2) ne détecte pas la zone de logo, le second facteur de rapport est appliqué.
Pour aller plus loin et partant de l'a priori que le logo est situé en haut du ticket de caisse, cette stratégie est appliquée uniquement sur la partie supérieure des images du ticket. La partie inférieure est traitée, seulement si aucun logo n'est détecté. Cette stratégie permet à la fois de réduire les coûts de calcul et de manière indirecte de détecter l'orientation des tickets.

Étape 2 : Segmentation Sémantique de l'image

Une fois que l'image a été prétraitée pour être dans les meilleures conditions et favoriser l'analyse, nous appliquons le réseau de segmentation « SS DCNN B » afin d'obtenir une carte de chaleur contenant les différentes zones de localisation du logo. La taille de la carte de chaleur varie d'une image à l'autre, puisque la taille d'entrée du réseau n'est pas fixe (contrairement au réseau « SS DCNN A ») mais elle est adaptée dynamiquement pour être conforme à la taille de l'image après pré-traitement, comme illustré sur la figure 4.24.

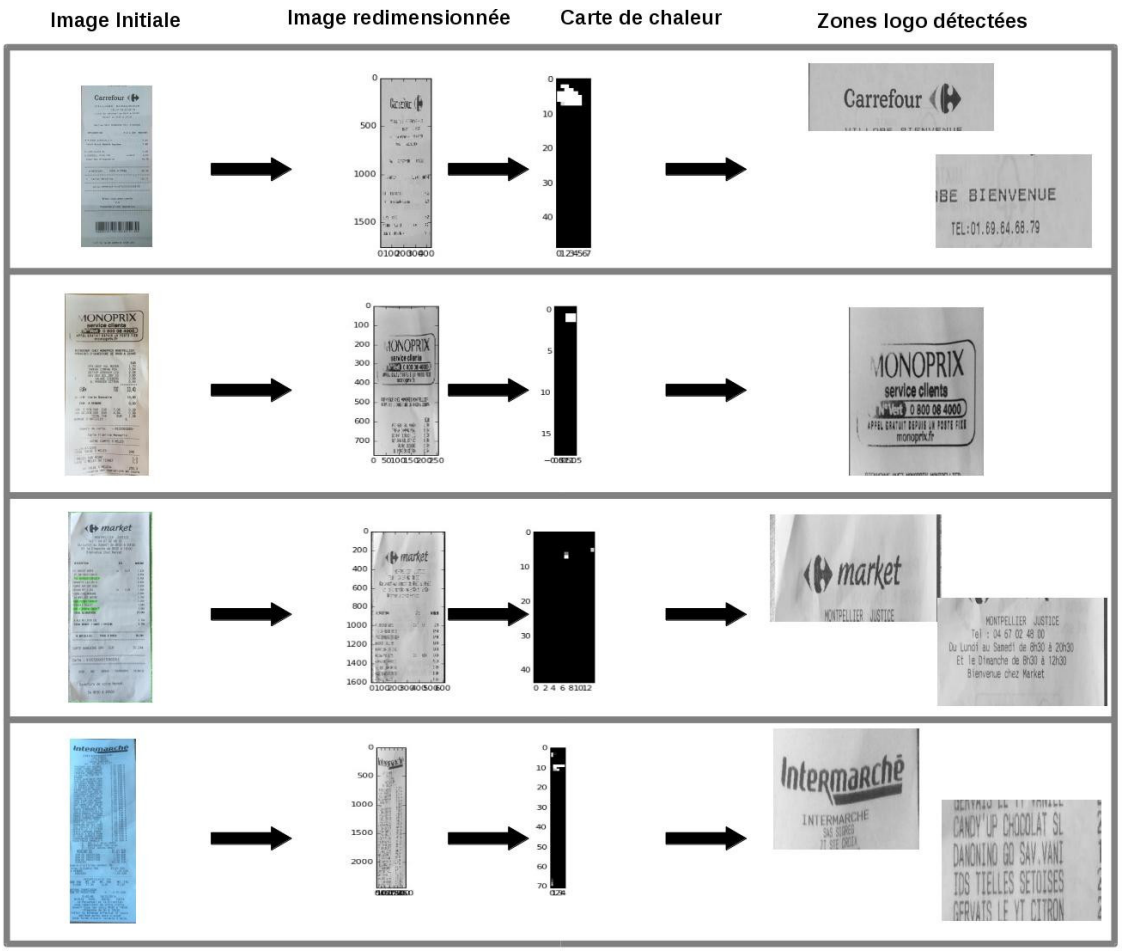


FIGURE 4.24. – Segmentation sémantique de l'image pour la localisation du logo avec l'utilisation du réseau de neurones « SS DCNN B ».

À partir de la carte de chaleur, plusieurs zones correspondant à des logos sont extraites. Seules les premières zones, celles situées en haut de la carte de chaleur sont prises en considération. Comme le ticket de caisse est un document structuré et que l'image est considérée comme étant correctement rognée, peu d'informations sont situées avant le logo. Les zones potentielles sont ensuite soumises au réseau de classification « Classif. DCNN B ». La seule zone retenue est celle qui obtient la plus grande probabilité de classification des enseignes, comme on peut le voir sur la figure 4.25, étape 2.
Cependant , à ce stade, la confiance dans la classification précise des enseignes n'est pas suffisante. En effet, la résolution grossière de la segmentation sémantique conduit à la définition d'une boîte englobant la zone du logo beaucoup trop large, qui peut inclure des informations voisines perturbatrices (bordures blanches ou texte) et donc pouvant affecter la précision de la classification du logo.

FIGURE 4.25. – Localisation et reconnaissance du logo.

Étape 3 : Classification du logo

Une dernière étape est donc nécessaire pour affiner ce résultat de classification préliminaire. Afin de supprimer les informations voisines et perturbatrices, le recadrage de la zone logo beaucoup trop large consiste en deux étapes très simples :

— binarisation de l'image ;

— projection des pixels.

La méthode utilisée ici, correspond exactement à l'approche décrite au chapitre 2 section 2.2.5. À l'issue du recadrage, plusieurs blocs de texte peuvent être générés. Chacun des blocs est à nouveau soumis au réseau de classification « Classif. DCNN B », le meilleur résultat (meilleure « précision » sans considérer la classe « non logo ») est retenu. Cette opération en deux étapes entraîne une augmentation significative non seulement des performances de classification, mais également des performances de localisation, comme illustrée sur la figure 4.25, étape 3.

Finalement on retient deux valeurs d'enseignes issues de l'analyse d'image, le « Top-1 » et le « Top-2 », identifiées par la dernière classification du réseau « Classif. DCNN B ».

Les performances obtenues pour la reconnaissance de l'enseigne via l'analyse d'image sont présentées plus loin dans la section 4.5.2.

4.5.1.b. Analyse du texte

La deuxième méthode représentée sur la figure 4.22, permettant la détection de l'enseigne repose également sur une analyse de texte. L'image en sortie de la deuxième étape du pré-traitement « Localisation du ticket », rognée et redressée est soumis à nouveau à un outil de Reconnaissance Optique de Caractères : GoogleVision [21].

La qualité de la lecture réalisée par l'OCR est bien meilleure cette fois-ci, car l'image est débarrassée de son arrière-plan, mais elle n'est toujours pas assez précise pour assurer une analyse complète du ticket. Elle est en revanche suffisante pour reconnaître l'enseigne

Pour assurer la reconnaissance de l'enseigne à partir de la sortie de l'OCR, trois critères sont recherchés.

Critère n°1 : le nom de l'enseigne :
Le premier critère est basé tout simplement sur le nom du magasin. Pour cela, nous avons manuellement constitué une base de données des noms des différentes enseignes. Cette base contient à ce jour plus de 200 enseignes, avec environ 70 grandes enseignes nationales, telles que « Auchan », « Carrefour », « Géant Casino », etc. Ces dernières correspondent aux enseignes que l'on rencontre le plus souvent (les 17 enseignes sélectionnées pour l'analyse de logo appartiennent à cette catégorie). Un nom de magasin connu est donc recherché dans le texte lu sur le ticket de caisse. Afin de compenser les erreurs de l'OCR au niveau des caractères, l'approche des N-gramme avec N = 2 est considérée. En fin de compte, le nom du magasin ayant la mesure de confiance la plus grande est retenu.

Critère n°2 : le numéro de téléphone :
Le second critère consiste à identifier la présence d'un numéro de téléphone en utilisant une expression régulière. S'il est détecté, ce numéro de téléphone est comparé à ceux d'une base de données de numéros de téléphone de magasins connus. Nous avons manuellement constitué cette base de numéros de téléphone de magasins et nous continuons à l'enrichir. Si un numéro détecté ne figure pas dans notre base, une requête est faite dans un annuaire inversé en ligne. D'une part cela nous permet de retrouver l'enseigne, et d'autre part d'enrichir notre base de données.

Critère n°3 : la terminologie de l'enseigne :
Enfin, le troisième critère est basé sur des références à la terminologie de l'enseigne. Pour chaque enseigne, la terminologie utilisée en termes de slogan, de programme de fidélité et de distributeur de marques a été répertoriée par l'entreprise (un exemple est présenté dans le tableau 4.8). Les termes étant spécifiques à chaque enseigne, trouver un tel terme rend possible l'identification de l'enseigne.

Le résultat final est une liste de paires dont le premier élément est l'enseigne du magasin identifié et le second élément est un poids qui représente le nombre de critères (0,1,2 ou 3) qui ont permis d'identifier cette enseigne. Cette liste peut être vide, surtout lorsque le ticket de caisse n'est pas entièrement contenu dans l'image ou si la sortie de l'OCR n'est pas suffisamment riche.

Terminologie de l'enseigne « Auchan »	
Carte de Fidélité	Waaoh ! ! !
Slogan	**Marques Distributeur**
"Vivons mieux. Vivons moins cher" "La vie, la vraie" "La vie Auchan, elle change la vie" "Et vous la vie vous l'aimez comment ?"	Auchan, Mmm, Mieux vivre Pouce, Rik&Rok, In extenso, Selecline, Cup's, Cosmia, Qilive, Baby

TABLE 4.8. – Exemple de la terminologie d'une enseigne (Auchan).

4.5.1.c. Fusion

Nous venons de présenter deux méthodes permettant de déterminer l'enseigne du ticket, une basée sur l'analyse d'image puis une deuxième sur l'analyse de texte. En plus de la détection du ticket, l'analyse d'image permet la détection de l'orientation du ticket (inversion haut / bas) en supposant que le logo de l'enseigne est situé en haut du document.

La dernière phase de cette étape de pré-traitement consiste à fusionner les résultats des deux algorithmes, afin d'extraire l'enseigne du ticket. Le nom de l'enseigne est associé automatiquement à un ticket de caisse dans l'une des situations suivantes :

— les trois critères de l'analyse textuelle fournissent le même résultat. Dans ce cas l'analyse d'image n'est pas utilisée ;
— une des deux enseignes déterminées par l'analyse d'image (« Top-1 », « Top-2 ») correspond à une enseigne fournie par au moins deux critères de l'analyse de texte.

Dans les autres cas, l'analyse automatique est trop incertaine compte tenu des fortes contraintes industrielles et la détermination de l'enseigne à cette étape est redirigée vers l'analyse experte d'un humain.

L'intérêt de la détection de l'enseigne pendant la phase de pré-traitement est qu'elle peut faciliter l'étape de segmentation de blocs de texte. En effet, les tickets de caisse étant des documents structurés, une fois l'enseigne du magasin connue, nous pouvons utiliser des connaissances a priori telles que la structure du ticket.

Nous présentons les résultats obtenus dans la section suivante 4.5.2.

4.5.2. Résultats et Performances

Les performances des deux algorithmes pris indépendamment ainsi que la performance obtenue après fusion sont reportées dans le tableau 4.10.

Class	Accuracy		IoU
	Top-1	**Top2**	
Carrefour	97%	100%	0.80
Market	100%	100%	0.85
Geant	92%	96%	0.70
Casino	97%	100%	0.70
Monoprix	100%	100%	0.80
Lidl	40%	60%	0.45
Leclerc	70%	90%	0.55
SuperU	90%	94%	0.75
Intermarche	96%	100%	0.80
Grand frais	90%	98%	0.65
Performance Globale	**87,2%**	**93,8%**	**0.7**

TABLE 4.9. – Les performances de localisation de la méthode d'analyse d'image sur un échantillon d'une dizaine d'image de ticket par enseigne.

La base d'évaluation est constituée d'environ 1000 images avec une soixantaine d'exemples par enseigne.

Comme montré ci-après, la fusion entre l'approche « analyse d'image » et « analyse de texte » améliore les performances globales et nous permet d'atteindre une précision intéressante de **97,7%**.

Nous avons également mesuré les performances de localisation réalisées par l'analyse d'image sur une dizaine d'enseignes, en utilisant la métrique IoU. Nous avons rapporté les résultats sur le tableau 4.9, ces performances étant associées aux performances de classification pour chacune des enseignes.

(a) IoU = 0.97 (b) IoU = 0.67 (c) IoU = 0.45

FIGURE 4.26. – Évolution de la valeur de la métrique IoU dans différentes configurations, en bleu clair la vérité terrain et en bleu foncé la détection faite par l'algorithme.

Reconnaissance de l'enseigne			
DCNN		**Text Sem.**	**DCNN & Text Sem.**
Top-1	**Top-2**		
89.9%	95.8%	80.7%	**97.7%**

TABLE 4.10. – Performance de détection de l'enseigne sur 17 enseignes.

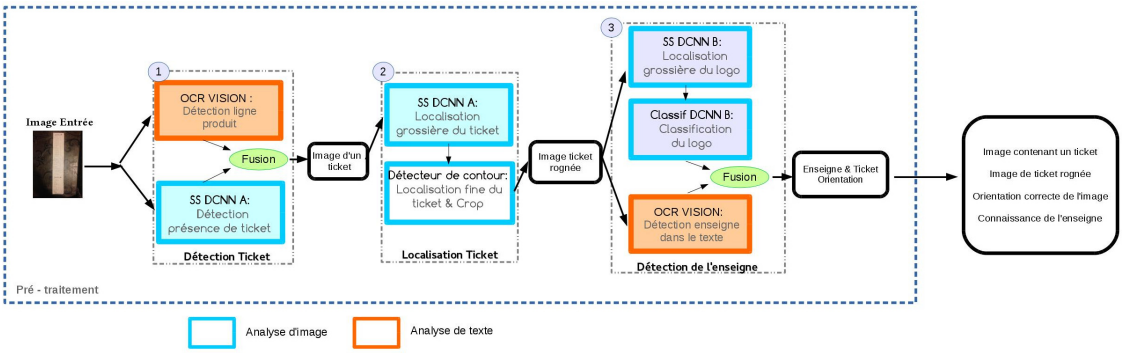


FIGURE 4.27. – Chaîne de traitement pour le pré-traitement d'analyse d'un ticket.

On peut noter que plus la valeur « Intersection sur Union » est élevée, plus la classification est bonne. On observe que la métrique « IoU » est une mesure métrique stricte, comme illustrée sur la figure 4.26 .

En ce qui concerne l'approche « analyse d'image », nous constatons que les petits logos (« Lidl » et « E.Leclerc ») et ceux qui sont situés près des bords du ticket de caisse sont les cas les plus difficiles qui affectent la performance et sont les sources d'erreur. Cependant ces situations sont des échantillons rares dans l'ensemble de données de test. En ce qui concerne l'analyse textuelle, les erreurs constatées sont dues à un mauvais fonctionnement de l'OCR lié à des images de mauvaise qualité ou à des graphismes un peu trop spécifiques.

4.6. Synthèse et conclusion

Dans ce chapitre, nous nous sommes intéressée à la réalisation de la première phase de la chaîne de traitement envisagée pour l'analyse automatique d'une image de ticket de caisse. Cette première phase est une phase de pré-traitement qui permet de traiter l'image brute et de ne retenir que les informations nécessaires à une analyse de bonne qualité. Nous avons résumé le déroulement de cette phase, avec les différentes approches retenues sur la figure 4.27.

À partir d'une image acquise avec un téléphone mobile, l'approche proposée permet

de filtrer les images ne contenant pas de ticket de caisse, de localiser le ticket s'il est présent dans l'image et de classer et localiser le logo de l'enseigne.

Lors de la phase de pré-traitement, nous avons essayé pour chacune des étapes d'introduire plusieurs méthodes mises en compétition. Ces méthodes assez spécifiques ont pour objectif d'apporter des informations sur chacune des données à extraire, à partir de sources différentes. En effet, un soin particulier est apporté à la fiabilité de chaque information extraite. Ainsi, autant que possible, des approches concurrentes et complémentaires sont considérées et fusionnées. D'un côté, nous profitons des réseaux de neurones formés pour la classification et de la segmentation sémantique aussi que des approches classiques d'analyse d'image pour une meilleure compréhension du contenu de l'image. D'un autre côté, nous utilisons une analyse sémantique textuelle basée sur la terminologie et la linguistique afin d'interagir avec un OCR en corrigeant ses erreurs et comprendre le sens du texte pour en extraire les informations recherchées.

Nous avons donc proposé un processus comprenant des combinaisons originales de méthodes et de techniques existantes adaptées à notre cas d'étude, aboutissant à de très bonnes performances.

5. Traitement : Lecture et analyse du contenu du ticket

Dans ce chapitre nous présentons les approches envisagées et évaluées pour les dernières étapes de la lecture automatique des tickets de caisse. Il s'agit de la localisation précise des différentes zones de texte, ainsi que la lecture du texte et l'analyse sémantique du texte ainsi extrait. Nous présentons donc les méthodes envisagées pour chacune des étapes ainsi que les performances obtenues.

5.1. Introduction

Dans ce chapitre, nous nous intéressons à l'analyse du contenu du ticket de caisse. Cette analyse est constituée de trois étapes, ayant respectivement pour objectif d'extraire les différents blocs de texte dans l'image de ticket de caisse, de lire le contenu à l'aide d'un outil de reconnaissance optique de caractère et ensuite d'interpréter le contenu afin d'extraire les informations recherchées. À partir des informations extraites, plusieurs objectifs sont recherchés. Dans un premier temps, il est important de pouvoir faire un retour à l'utilisateur comme par exemple lui apporter une aide utile à la gestion de son budget, et ce le plus rapidement possible. Dans un deuxième temps, ces données doivent permettre d'obtenir des statistiques de consommations. Associées à un profil d'utilisateur (âge, étudiant, mère de famille, etc ...) ces données ont une forte valeur ajoutée comme nous l'avons précisé au chapitre 1.

L'évaluation du démonstrateur, décrit au chapitre 2, nous a permis de mettre en évidence que la détection du texte est satisfaisante, mais doit gagner en robustesse lorsque l'acquisition n'est pas maîtrisée. Concernant la lecture du texte, les OCRs que nous avons testés ont été mis en défaut dans un certain nombre de situations. Le traitement en amont, proposé au chapitre 4 doit permettre de mettre en relief le contenu des blocs de texte afin d'assurer une bonne qualité de la reconnaissance de caractères. Quant à l'analyse sémantique, les difficultés se situent sur la présence d'informations ambiguës. La désambiguïsation n'est possible qu'au travers d'une délimitation et catégorisation précise des blocs de texte dans le ticket.

Dans l'objectif d'assurer une bonne qualité d'analyse, nous proposons la chaîne de traitement suivante constituée de trois étapes :

1. segmentation des blocs de texte ;

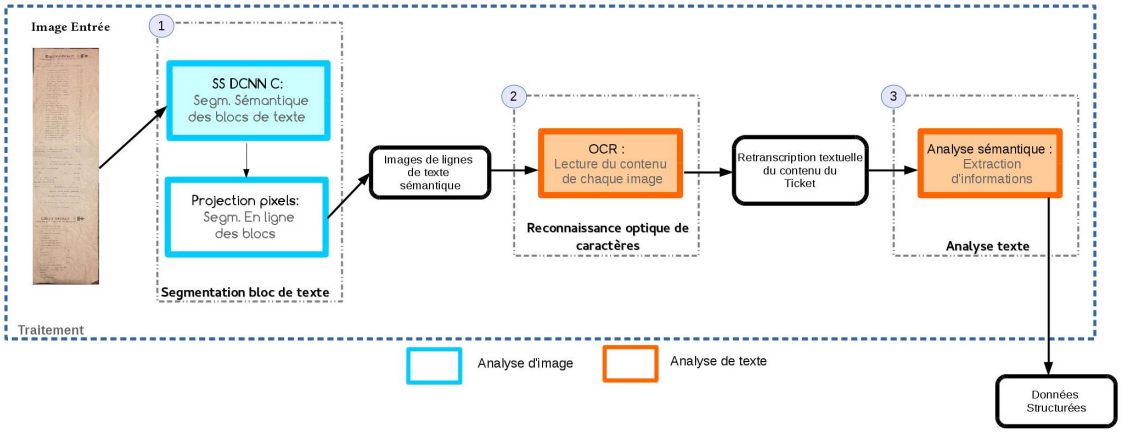


FIGURE 5.1. – Chaîne de traitement pour l'analyse du contenu d'un ticket de caisse après pré-traitement.

2. lecture du texte ;
3. analyse du contenu textuel.

Nous allons présenter en détail les différentes étapes de cette phase d'analyse dont l'organisation est résumée sur la figure 5.1.

5.2. La chaîne de traitement proposée

5.2.1. La localisation des zones de texte

À la sortie de la dernière étape de pré-traitement, nous disposons d'une image limitée au seul ticket, redressée si nécessaire, et nous avons connaissance de l'enseigne. Donc, dans la première étape de la chaîne de traitement, nous nous intéressons à la segmentation en blocs de zones de texte. Cette étape enchaîne deux opérations. La première opération, effectuée avec un réseau profond, réalise une segmentation sémantique du ticket en blocs afin de séparer l'entête, le logo, la liste de produits, la liste les prix, etc. L'intérêt est double. D'une part, cela permet de préparer la segmentation des lignes de textes qui va suivre, et, d'autre part, la catégorisation des blocs de texte permettra d'aider la correction des erreurs de lecture de l'OCR (par exemple, si dans la liste des prix l'OCR a reconnu « 4O », la correction de la lettre « O » par le chiffre « 0 » est naturelle). La seconde opération de cette étape est la segmentation en lignes effectuée par projections horizontales. La segmentation en lignes permet d'améliorer les performances de l'OCR et de mieux maîtriser la lecture.
Le bloc « 1 » de la figure 5.1 illustre cette approche qui constituera l'essentiel de ce chapitre.
À noter que cette phase de localisation des blocs de texte sera également évaluée en utilisant l'image dite dite « brute » (i.e non rognée et non redressée). L'objectif est d'évaluer l'apport de la phase de pré-traitement.

5.2.2. La lecture du texte

La deuxième étape est la reconnaissance de texte afin de traduire toutes les images de lignes de texte en caractères ASCII compréhensibles par une machine. La reconnaissance optique des caractères étant un domaine très actif depuis plusieurs décennies, plusieurs solutions opérationnelles sont disponibles aujourd'hui et offrent des performances très satisfaisantes dans des conditions favorables. On peut citer Tesseract [19] et Google Vision [21]. Nous avons donc envisagé l'utilisation de tels OCRs.
Cependant, leur utilisation présente un double inconvénient : d'abord, leurs performances sont limitées par le fait qu'ils ne sont pas optimisés pour des tickets de caisse. Ensuite, du point de vue de l'entreprise, certains de ces OCRs sont payants et leur utilisation doit donc être optimisée pour réduire les coûts et répondre aux contraintes d'usages (nombre de requêtes par seconde, etc ...). À cause de ces contraintes, nous avons donc exploré la piste de la construction d'un OCR dédié aux tickets de caisse. Pour cela, nous avons évalué plusieurs architectures de réseaux profonds pour la classification de caractères. La méthode proposée comporte deux étapes : dans un premier temps, chaque image de ligne de texte est segmentée en caractères afin d'être, dans un deuxième temps, soumise à un réseau de classification de caractères pour permettre ensuite de reconstituer le texte contenu dans les tickets.

5.2.3. L'analyse sémantique du texte lu

Enfin, dans une dernière étape, nous aborderons l'analyse sémantique du texte lu. Cette analyse sémantique recouvre de nombreux aspects qui débordent du cadre de cette thèse, et nous nous limiterons à quelques uns de ces aspects permettant d'extraire les informations nécessaires à l'interprétation du contenu du ticket. Par exemple, dans l'entête et dans le bas du ticket, nous allons rechercher les informations « date », « heure d'achat », « ville » et « nom du magasin », alors que dans le bloc contenant la liste des produits achetés nous allons rechercher l'interprétation des libellés courts en libellés longs.

Cette étape est réalisée à partir d'une source d'information imparfaite : le texte extrait suite à une succession d'étapes d'analyse d'images. La qualité du texte extrait est donc dépendante des performances de l'analyse d'image qui précède, mais aussi de la performance de l'outil de reconnaissance de caractères. Il est donc nécessaire de prendre en compte les éventuelles erreurs, omissions ou ajouts de caractères, manques d'informations, etc.

Nous avons donc commencé à mettre en place des outils d'analyse sémantique comme les lexiques et les terminologies. Un long travail d'enrichissement des connaissances a été réalisé, ainsi qu'une réflexion sur la construction d'une ontologie.

Nous allons présenter dans la suite, les approches proposées ainsi que les résultats obtenus pour ces trois étapes de l'analyse du contenu du ticket de caisse.

FIGURE 5.2. – Exemples d'images après pré-traitement. Les images sont redressées et débarrassées de leur arrière-plan.

5.3. La localisation des zones de texte

En ce qui concerne la segmentation des blocs de zones de texte, plusieurs solutions s'offrent à nous. La première solution, et sans doute la plus simple, est d'utiliser la méthode de projections présentée au chapitre 2 section 2.2.5. Cette méthode est efficace quand les conditions d'acquisitions sont correctes, comme nous avons pu le constater dans la mise au point du démonstrateur.

Cependant, nous souhaitons obtenir ici plus qu'une simple segmentation des blocs de texte comme nous l'avons présenté au paragraphe 5.2.1. L'objectif est d'obtenir une catégorisation des différents blocs de texte en fonction de leur contenu sémantique et de leur position dans le ticket. L'intérêt ici est d'apporter une solution au problème soulevé au chapitre 2 section 2.3.5 concernant l'ambiguïté de certains termes dans le ticket de caisse en introduisant un a priori sur le type de bloc de texte analysé (par exemple l'ambiguïté sur le terme « Orange » qui peut désigner la ville ou le fruit). Ainsi l'analyse sémantique appliquée par la suite pourra être différente en fonction de la nature du bloc.

Pour satisfaire l'ensemble de ces besoins, l'approche envisagée est une segmentation sémantique par un réseau de neurones profond, qui assignera chaque pixel à un bloc de texte, dont le type (entête, produits etc.) sera prédit.

Dans l'analyse de cette segmentation, nous envisagerons deux types d'images. Les premières sont les images dites « rognées », issues du pré-traitement, qui ne comportent donc que le ticket, éventuellement redressé. Mais nous testerons également nos méthodologies de segmentation sur les images dites « brutes » (i.e non rognée et non redressée). Cette stratégie a pour but de comparer les performances concernant la localisation du ticket réalisée lors de la phase de pré-traitement avec les méthodes spécifiques

de segmentation par patchs développées dans ce chapitre. L'utilité de la phase de prétraitement n'est pas pour autant remise en cause, car cette phase assure de la présence d'un ticket, détecte l'enseigne, vérifie l'orientation du ticket grâce à la détection du logo et fournit une localisation grossière du ticket dans l'image.

5.3.1. Base d'apprentissage

Pour régler les réseaux de neurones profonds de façon supervisée, nous avons constitué une base d'apprentissage à partir des images de tickets recueillis par l'entreprise (voir chapitre 4 : 4.2.3).

5.3.1.a. Annotation

Nous avons défini manuellement pour chaque image de ticket de la base les différentes zones utiles en distinguant un maximum de 9 classes (dans le cas d'utilisation des images brutes), comme illustrée sur la figure 5.3 :

— fond *classe présente uniquement dans le cas où les images brutes sont utilisées* ;
— ticket ;
— logo ;
— entête ;
— libellé produit ;
— prix ;
— bas de ticket ;
— code barre ;
— autre texte.

Pour cela, la plateforme Web développée au chapitre 4 pour l'annotation des premières bases d'apprentissage (section 4.2.3.b) a été adaptée afin d'annoter les différentes zones. Leur boîte englobante est définie par au moins quatre sommets définissant un polygone avec la contrainte que les polygones englobant des zones de texte ne peuvent se chevaucher.

Ainsi nous avons constitué une base d'apprentissage contenant environ 950 images. La base couvre au total 19 enseignes avec une cinquantaine d'exemples par enseigne. Cette séparation permet d'obtenir une base de données équilibrée et diversifiée.

L'annotation est conduite de la façon suivante :

La zone de l'image contenant le ticket de caisse (classe « Ticket de caisse ») est d'abord isolée de l'arrière plan. Ensuite dans le ticket, tout texte au-dessus du bloc de produit est considéré comme faisant partie de la classe « entête », sauf le « logo » et le « code barre » qui constituent des classes spécifiques. S'il existe des zones de texte séparées de l'entête ne contenant aucune information sémantique intéressante, celles-ci

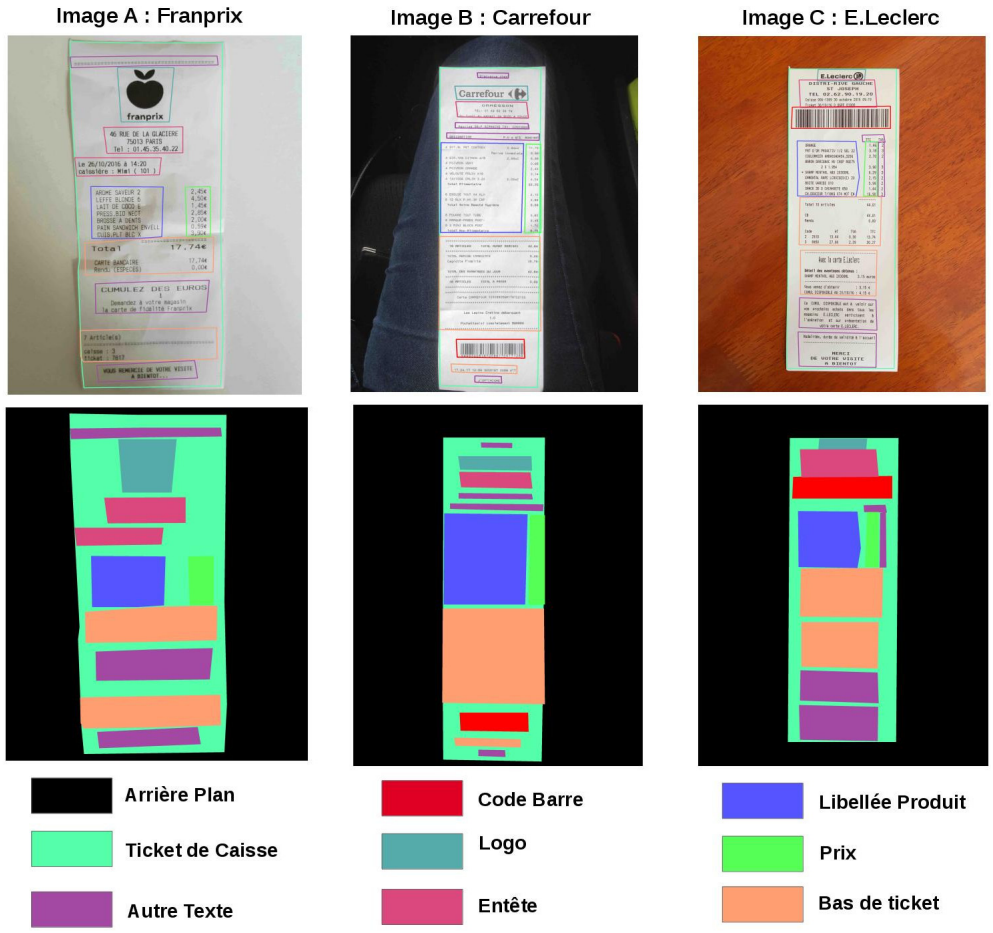


FIGURE 5.3. – Illustration des images obtenues après annotation pour la segmentation sémantique. La première ligne montre les différentes boîtes englobantes définies manuellement, sur la deuxième ligne sont représentées les cartes de segmentation qui en résultent et qui serviront à l'apprentissage.

sont considérées comme appartenant à la classe « Autre Texte », (par exemple le texte « Bienvenue chez » sur les tickets de Carrefour, voir figure 5.3, *image "B"*). L'entête peut être séparé en plusieurs blocs distincts si l'espace entre des lignes ou des groupes de lignes est trop important, comme on peut le voir sur la figure 5.3, *image "A"*. L'idée est donc d'éviter l'introduction de zones blanches trop grandes à l'intérieur d'une classe. Nous procédons de la même manière pour le texte en dessous du bloc de produit, appartenant donc à la classe « Bas de ticket ». Ici également il y a une distinction avec le bloc contenant des informations sémantiquement intéressantes, comme le montant total, le nombre d'articles, ou encore la date d'achat (classe « Bas de ticket ») et les autres blocs de texte (classe « Autre Texte »). Cependant, cette distinction ne sera pas forcément prise en compte pendant la phase d'apprentissage, car il paraît difficile de les différencier au niveau du pixel. Enfin le bloc « libellé produit » et le bloc « prix » sont séparés.
De manière très simple, cette annotation en 9 classes pourra éventuellement être réduite à un jeu de classes plus grossier comme nous le verrons par la suite.

5.3.2. Les Architectures utilisées

Nous nous sommes inspirée, dans un premier temps, de deux architectures présentées dans la littérature : ENet [60] et FC-DenseNet [61], toutes deux utilisées pour réaliser de la segmentation sémantique. Nous avons présenté ces deux architectures au chapitre 3 section 3.2.8. Leurs complexités sont très différentes et leur comparaison permettra de juger de la difficulté du problème.

5.3.2.a. Segmentation sémantique

Architecture ENet : La première architecture est ENet présentée dans l'article [60]. L'intérêt de cette architecture est d'être capable de fonctionner sur des appareils peu puissants tels que les appareils mobiles tout en atteignant des performances équivalentes au réseau SegNet [56]. La figure 3.9 au chapitre 3 décrit l'architecture complète du réseau inspiré de l'architecture ResNets [38].

L'architecture originale de ce réseau contient 370 000 paramètres. Cependant dans nos expérimentations nous avons mis en place des architectures moins profondes (voir section suivante) afin de réduire le nombre de paramètres à définir. Notre base d'apprentissage étant très petite, cela peut être un facteur très influent. Nous avons utilisé comme critère d'optimisation / fonction de « Loss » la « cross-entropy ».

Architecture FC-DenseNet : L'architecture FC-DenseNet présentée dans l'article [61] est représentée sur la figure 3.12 au chapitre 3.

Dans l'article [61], le réseau évalué est constitué de cinq blocs denses de taille croissante pour la partie encodeuse et le même nombre de blocs denses pour la partie décodeuse, ce qui représente un total de 103 couches de convolutions.

Comme pour la première architecture dans nos expérimentations nous avons utilisé des réseaux s'inspirant de ce modèle, mais moins profonds toujours dans le but de réduire le

nombre de paramètres et d'améliorer les performances de segmentation du réseau avec notre base d'apprentissage. Nous allons présenter dans la section 5.3.3 les différentes expériences que nous avons menées avec différents niveaux de segmentation (localisation du ticket, localisation des blocs de texte, localisation du logo ou segmentation des 9 classes de blocs de texte). Comme pour la première architecture le critère d'optimisation est la fonction de « cross-entropy ».

5.3.2.b. Segmentation sémantique multi-tâche

À partir de ces approches de l'état de l'art, nous proposons une architecture de segmentation sémantique multi-tâche originale. Cette architecture va générer plusieurs sorties répondant chacune à une tâche précise. L'avantage de l'apprentissage multi-tâche est de pouvoir exploiter les points communs entre les tâches et de rendre ainsi le réseau plus performant sur chacune des tâches.

Les architectures que nous avons proposées s'inspirent du réseau FC-DenseNet présenté au chapitre 3. Dans la figure 5.4, nous avons illustré un exemple de réseau multi-tâche basé sur ce réseau.

Ces architectures varient en fonction des objectifs à atteindre. Les variations se situent naturellement sur le nombre de tâches apprises simultanément par le réseau, mais également sur la profondeur du réseau. Dans l'illustration 5.4, le réseau est conçu pour l'apprentissage simultané de deux tâches (détection des bords du ticket et détection du ticket) et la profondeur est augmentée d'un seul bloc dense « encodeur » (ce qui impose l'ajout de deux autres blocs denses : le bloc dense central et le bloc dense « décodeur »), de taille variable. Cependant, afin de rester fidèle à l'architecture du réseau FC-DenseNet original, nous avons respecté la profondeur des blocs denses de l'architecture présentée dans [61] et décrit au chapitre 3 dans le tableau 3.1. Pour rappel l'architecture FC-DenseNet proposée dans [61] comporte cinq blocs denses (DB) avec un nombre de couches de convolution différent et croissant :

— Bloc Dense 1 : 4 couches ;
— Bloc Dense 2 : 5 couches ;
— Bloc Dense 3 : 7 couches ;
— Bloc Dense 4 : 10 couches ;
— Bloc Dense 5 : 12 couches.

La stratégie que nous allons adopter ici est intuitivement d'attribuer le chemin le plus court, c'est-à-dire le chemin avec le moins de blocs denses, à la tâche la plus simple (le plus bas niveau). Plus la tâche est complexe (plus haut niveau), plus le chemin sera long avec un nombre de couches plus important. Concernant le critère d'optimisation nous avons ici fait le choix d'attribuer le même poids à chaque tâche du réseau multi-tâche et donc de sommer tout simplement chaque « loss » de chaque tâche obtenue via la fonction de « cross-entropy ».

Nous allons maintenant présenter les spécificités des différentes architectures multi-tâches envisagées.

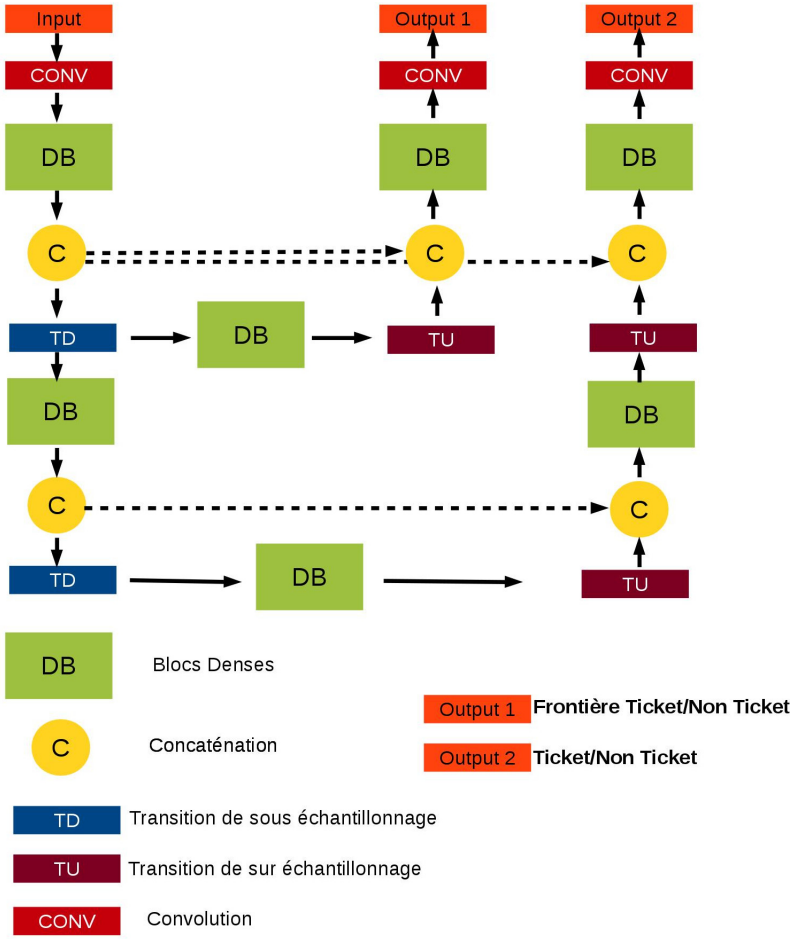


FIGURE 5.4. – Diagramme d'un exemple d'architecture multi-tâche basée sur le réseau FC-DenseNet, avec ici deux tâches différentes.

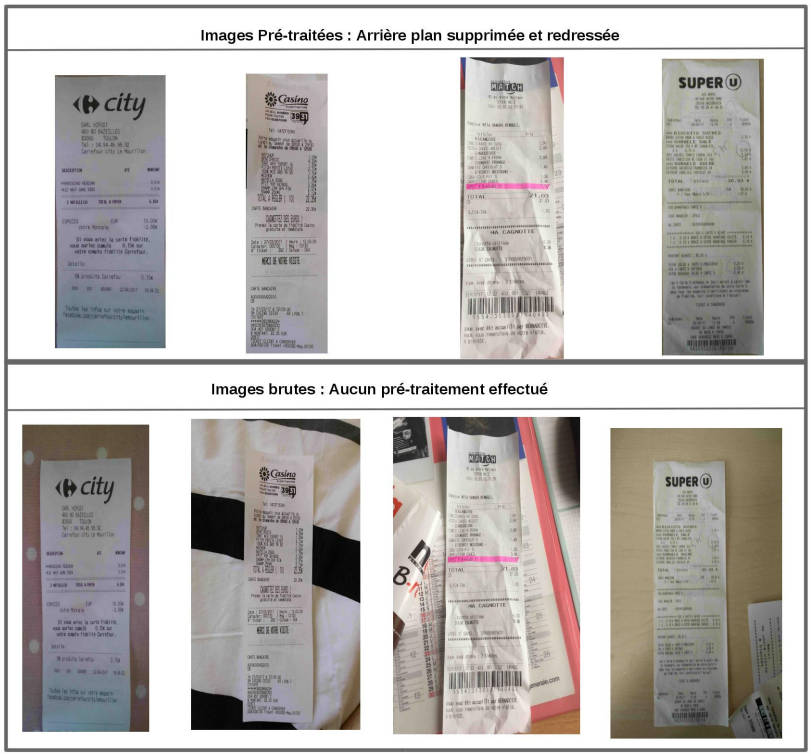


FIGURE 5.5. – Les deux différentes bases de données prises en compte pour la segmentation sémantique des blocs de texte. La première base a été pré-traitée (chapitre 4) alors que la deuxième base correspond aux images brutes, éventuellement redressées.

5.3.3. Mise en œuvre et performances

La mise en œuvre des approches proposées a été effectuée avec le framework « TensorFlow » en envisageant deux situations comme cela a été évoqué en introduction à ce chapitre :

1. on considère l'image issue de la phase de pré-traitement, ce qui correspond à une image rognée et redressée ;
2. on considère l'image brute envoyée par le serveur, qui grâce à la phase de pré-traitement, contient un ticket dans la bonne orientation et localisé grossièrement. La seule différence est qu'ici l'image n'a pas été rognée.

La figure 5.5 illustre les images d'entrée utilisées dans ces deux situations.

Les tâches considérées, directement dépendantes du nombre de classes envisagées, sont présentées dans le tableau 5.1.

Nous avons mis en place une série de tests permettant d'évaluer l'architecture la plus adaptée à notre situation entre ENet et FC-DenseNet. Ces deux architectures sont

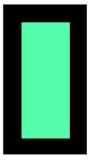
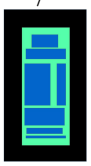
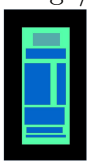
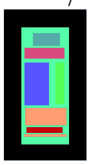
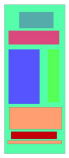
Images Brutes	**Images Pré-traitées**
Localisation du ticket	
Problème à deux classes	
Fond / Ticket 	
Localisation des blocs de texte	
Problème à trois classes	Problème à deux classes
≪ Fond / Ticket / Blocs de texte ≫ 	≪ Ticket / Blocs de Texte ≫ 
Localisation du logo et blocs de texte	
Problème à quatre classes	Problème à trois classes
≪ Fond / Ticket / Logo/ Blocs de texte ≫ 	≪ / Ticket / Logo/ Blocs de texte ≫ 
Localisation des différents blocs de texte	
Problème à huit classes	Problème à sept classes
≪ Fond/Ticket/Logo/Entête/Produits ≫ ≪ Prix/Bas Ticket/Code Barre ≫ 	≪ Ticket/Logo/Entête/Produits ≫ ≪ Prix/Bas Ticket/Code Barre ≫ 

TABLE 5.1. – Différentes configurations des problèmes de segmentation

ENet		ENet réduit		FC-DenseNet		FC-DenseNet réduit	
370 000		310 000		9 400 000		1 600 000	
Acc	**MeanIoU**	**Acc**	**MeanIoU**	**Acc**	**MeanIoU**	**Acc**	**MeanIoU**
0.61	0.30	0.59	0.40	0.53	0.21	0.62	0.42

TABLE 5.2. – Performances des différentes configurations des architectures ENet et FC-Densenet, sur un problème de segmentation à 8 classes, obtenues après 800 000 itérations, sur les images issues du pré-traitement (la deuxième ligne du tableau indique le nombre de paramètres de chaque réseau).

évaluées en considérant les indicateurs suivants : la métrique IoU sur la base de test, et l' « Accuracy » .

A noter que toutes les expérimentations ont été réalisées grâce à la mise à disposition par l'université Savoie Mont Blanc de la plateforme de calcul MUST.

5.3.3.a. Segmentation Sémantique

Nous avons dans un premier temps évalué les performances des architectures ENet et FC-DenseNet, en comparant les performances des réseaux originaux et des réseaux de même nature mais beaucoup moins profonds. Nous avons considéré le problème de segmentation le plus précis, à savoir la segmentation de l'image en 9 classes (voir tableau 5.1). La structure originale du réseau ENet est composée de 5 phases, une phase consistant en une succession de modules « Bottleneck » dont le nombre varie selon les phases (cf chapitre 3). Chaque module de « Bottleneck » comprend :

— une projection 1×1 qui réduit la dimension ;
— une couche de convolution principale , régulière, dilatée ou complète, 3×3 ;
— une extension 1×1.

Nous avons proposé une architecture moins profonde en supprimant tout simplement la troisième phase. Comme cette phase supprimée ne comporte pas d'opération de sous-échantillonnage, la structure avale reste identique à celle de l'architecture originale.
Quant au réseau FC-DenseNet, nous avons proposé une architecture avec seulement trois blocs denses dont le nombre de couches est respectivement 4, 5 et 7 au lieu des 5 blocs denses de l'architecture originale. La taille des « bottleneck » est de 15 et le taux de croissance de chaque couche (« growth rate ») est de 16.

Les performances obtenues avec ces quatre architectures, sur notre base d'apprentissage de segmentation sémantique de ticket de caisse (voir section 5.3.1), sont reportées dans le tableau 5.2 pour les images pré-traitées (i.e rognées et redressées) et dans le tableau 5.3 pour les images dites « brutes ». Les résultats sont obtenus au bout de 800 000 itérations (nombre d'itérations au bout duquel les performances ne varient plus) et un nombre d'« Epoch » à 50.

ENet		ENet réduit		FC-DenseNet		FC-DenseNet réduit	
370 000		310 000		9 400 000		1 600 000	
Acc	**MeanIoU**	**Acc**	**MeanIoU**	**Acc**	**MeanIoU**	**Acc**	**MeanIoU**
0.59	0.35	0.58	0.38	0.54	0.25	0.61	0.40

TABLE 5.3. – Performances des différentes configurations des architectures ENet et FC-Densenet, sur un problème de segmentation à 9 classes, obtenues après 800 000 itérations, sur les images brutes (la deuxième ligne du tableau indique le nombre de paramètres de chaque réseau).

L'observation des résultats des tableaux 5.2 et 5.3 permet d'abord de constater que la réduction de la profondeur des réseaux n'a pas le même effet sur les réseaux ENet et FC-DenseNet. Sur le réseau ENet, les performances ont tendance à être moins bonnes, alors qu'avec le réseau FC-DenseNet, il y a une amélioration sensible des performances. Ensuite, que ce soit dans le problème à 8 ou à 9 classes, c'est le réseau FC-DenseNet réduit qui obtient les meilleures performances. Ces deux constats s'expliquent très certainement par la taille de notre base d'apprentissage qui ne contient, rappelons-le, qu'un millier d'images. Le nombre important de paramètres des architectures originales FC-DenseNet et ENet les rend difficilement entraînables à partir de si peu de données. Le bon compromis est donc d'utiliser une architecture FC-DenseNet simplifiée afin de réduire le nombre de paramètres tout en gardant la flexibilité du réseau FC-DenseNet qui permet de mieux répondre à notre problème. La réduction du nombre de paramètres permet sûrement de réduire l'« overfitting » pendant la phase d'apprentissage. Enfin, en considérant la meilleure architecture (FC-DenseNet réduit), si on compare les résultats de la segmentation à 8 classes (images pré-traitées) avec ceux de la segmentation à 9 classes (images brutes), même si la comparaison directe est discutable car le nombre de classes est différent, on note une légère supériorité apportée par le pré-traitement.

Pour affiner l'examen de ces performances, nous avons calculé les matrices de confusions par classe (figure 5.6). On constate que les confusions entre les différentes classes sont importantes notamment entre la classe « Entête » et la classe « Produit ». On constate également que la classe « Autre Texte » est mal dissociée des autres classes. On peut également observer (deuxième colonne des matrices de confusions) que la classe « ticket » est trop prédite. Globalement, ces performances ne sont pas suffisantes (IoU maximum à 0.4) et nous allons donc envisager des architectures multi-tâches.

5.3.3.b. Évaluation des performances des réseaux multi-tâches

Comme décrit à la section 5.3.2.b, nous avons proposé une architecture FC-DenseNet multi-tâche. L'intérêt ici est d'évaluer si l'entraînement simultané d'un tel réseau sur différentes tâches permet un réel apport sur les performances des différentes tâches

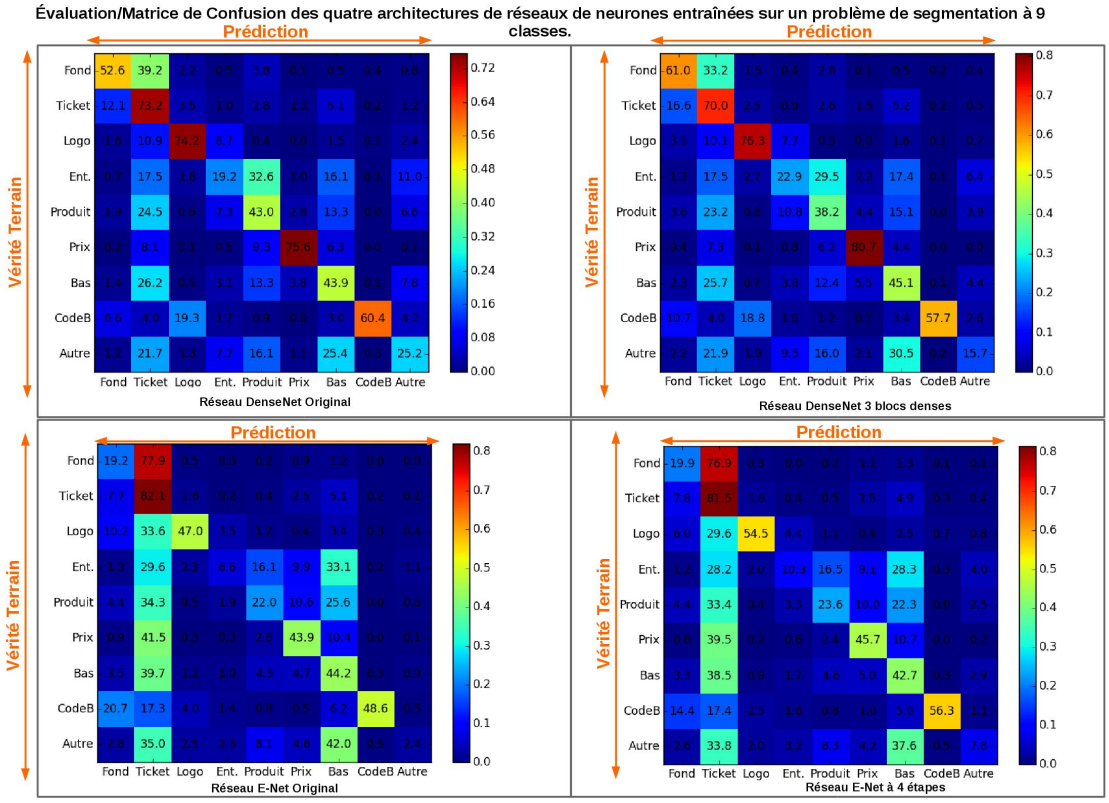


FIGURE 5.6. – Matrice de confusion des quatre architectures évaluées sur un problème de segmentation à 9 classes. En haut les deux architectures inspirées de FC-DenseNet et en bas les deux architectures inspirées de ENet.

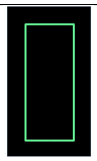
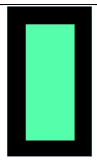
Sortie 1	Sortie 2
Détection des bords du ticket	Détection des pixels du ticket
	

TABLE 5.4. – Configuration du réseau multi-tâche pour la détection du ticket dans l'image

considérées individuellement.

Nous allons donc présenter trois expérimentations différentes et les résultats obtenus.

1ère expérimentation : Évaluation du multi-tâche sur un problème à deux classes

La première expérimentation que nous avons menée est l'évaluation des performances d'un réseau multi-tâche concernant la localisation du ticket dans l'image, donc un problème simple contenant deux classes de segmentation (cf la table 5.4). Pour cela, nous avons utilisé l'architecture présentée sur la figure 5.4 contenant deux parcours :

— un parcours court pour la tâche de détection des frontières entre les deux classes « Ticket » et « Fond » contenant un bloc dense de quatre couches de convolution (pour l'encodage), un « bottleneck » de 15 couches de convolution et un bloc dense de décodage ;

— un parcours long (deux blocs denses d'encodage, un « bottleneck » , deux blocs denses de décodage) pour la tâche de segmentation sémantique du « Ticket » et du « Fond ». La partie encodage est allongée d'un bloc dense de 5 couches de convolution, le bottleneck contient 15 couches de convolution.

Les deux parcours ont donc en commun le premier bloc dense de 4 couches de convolution.

Nous nous sommes intéressée uniquement aux performances de la deuxième sortie. Afin de comparer avec un apprentissage mono-tâche, nous avons entraîné dans les mêmes conditions (même base d'apprentissage, même machine...) un réseau monotâche dont l'architecture correspond au parcours long (tâche 2). Comme pour toutes les architectures multi-tâches que nous proposons ici, le critère d'optimisation est la fonction de « cross-entropy », sans pondération entre les différentes tâches. Nous avons reporté les résultats obtenus dans le tableau 5.5.

On constate que toutes les performances sont nettement meilleures concernant l'apprentissage en multi-tâche. Non seulement la précision de la métrique « IoU » est supérieure, mais la convergence du réseau est plus rapide. En effet, il a suffi de 200 000 itérations au réseau multi-tâche pour atteindre ces performances, tandis que le réseau

Type Entraînement	Nb itérations	BatchSize	Nb Epoch	MeanIoU	Accuracy
Mono-tâche	700 000	2	50	0.70	84%
Multi-tâche	**200 000**	**2**	**50**	**0.81**	**89%**

TABLE 5.5. – Résultats Mono-tâche et Multi-tâches sur problème à deux classes « Fond/Ticket »

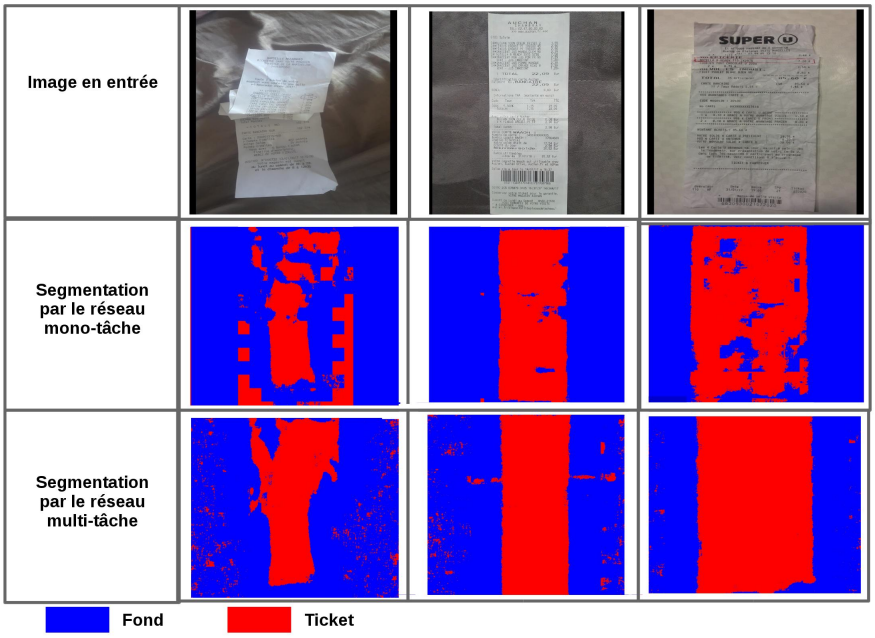


FIGURE 5.7. – Exemples de comparaison des cartes de segmentation obtenues avec le réseau mono-tâche (détection du ticket) et avec le réseau multi-tâche, à deux tâches (détection du ticket et localisation des bords du ticket).

mono-tâche a dû effectuer plus de 700 000 itérations. Sur la figure 5.7 est illustrée la différence des cartes de segmentation obtenues par l'apprentissage mono-tâche et par l'apprentissage multi-tâche. Les résultats obtenus par le réseau multi-tâche sont plus précis et la segmentation résultante illustrée sur la figure 5.8 le confirme.

2ème expérimentation : Évaluation des performances sur une architecture contenant trois sorties redondantes

De la même manière nous avons proposé une architecture de réseau multi-tâche (cf figure 5.9), avec trois tâches différentes mais comportant cette fois-ci une redondance :

— **Sortie 1** => Problème à deux classes : Fond/Ticket ;
— **Sortie 2** => Problème à trois classes : Fond/Ticket/Texte ;
— **Sortie 3** => Problème à quatre classes : Fond/Ticket/Texte/Logo.

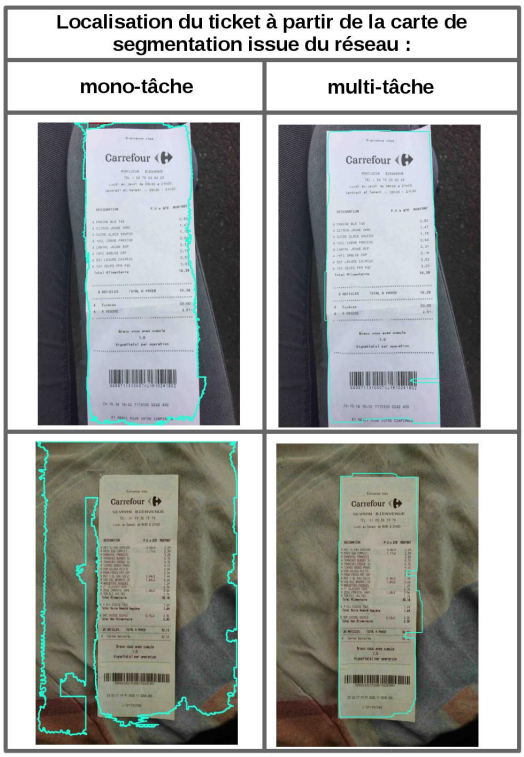


FIGURE 5.8. – Exemple de comparaison de la localisation du ticket réalisée à partir des cartes de segmentation, avec le réseau mono-tâche (détection du ticket) et avec le réseau multi-tâche à deux tâches (détection du ticket et localisation des bords du ticket).

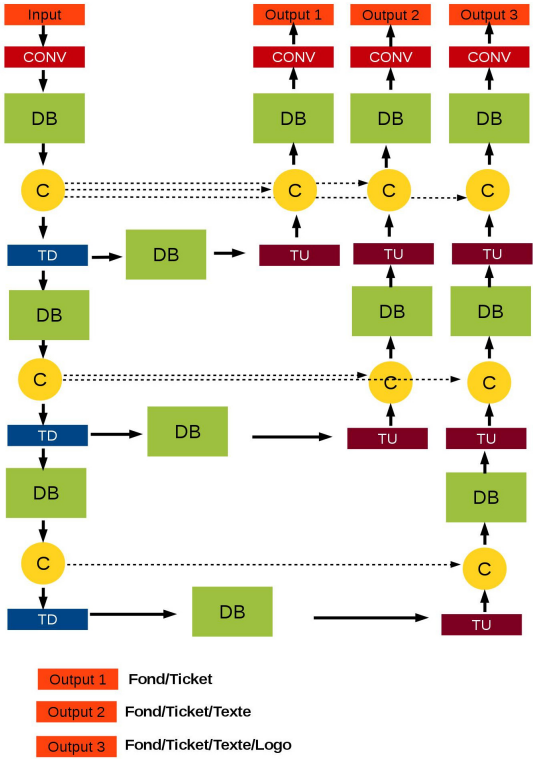


FIGURE 5.9. – Architecture multi-tâche basée sur le réseau FC-DenseNet, contenant trois sorties redondantes.

Les trois sorties sont dites redondantes, car la tâche réalisée par la première sortie est également réalisée par la sortie suivante tout en ajoutant de la précision sur le niveau de segmentation. On augmente systématiquement de « 1 » le nombre de classes dans la carte de segmentation à chaque sortie.
Parallèlement nous avons défini trois architectures mono tâche correspondant à chacune des trois sorties. Afin que les comparaisons aient du sens, la structure mono-tâche correspond au chemin de la tâche considérée dans l'architecture multi-tâche. Le nombre de blocs denses est identique pour chaque tâche. Les performances sont reportées sur le tableau 5.6. Sur ce tableau, on constate que plus on rajoute de classes, plus les performances baissent, ce qui s'explique par le fait que la difficulté du problème croît avec le nombre de classes.

Encore une fois les performances sont systématiquement meilleures dans la configuration en multi-tâche , aussi bien concernant la valeur de la métrique « IoU » qu'au niveau du nombre d'itérations nécessaires pour que le réseau converge. On note également que la différence de parcours entre la première et la deuxième expérimentation pour la tâche « Fond/Ticket » a un net impact sur les performances de segmentation (MeanIoU à 0,81 dans la première expérimentation et MeanIoU à 0,78 dans la deuxième

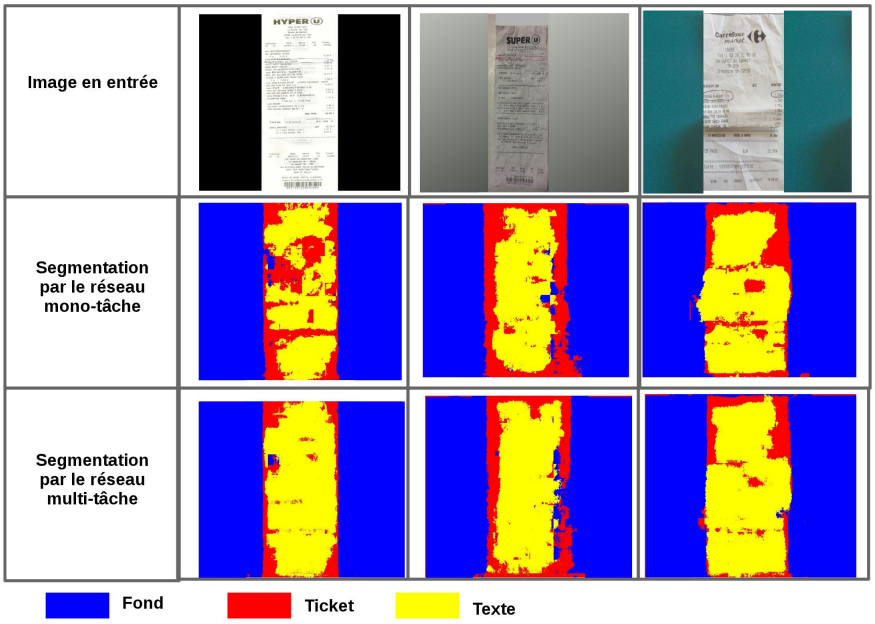


FIGURE 5.10. – Comparaison des cartes de segmentation pour le problème à 3 classes (Fond/Ticket/Texte) réalisées d'abord par le réseau mono-tâche et puis par la dernière sortie du réseau multitâche.

expérimentation). En effet, dans la première expérimentation la tâche « Fond/Ticket » a un parcours plus long : deux blocs denses d'encodage contre un unique bloc dans la deuxième expérimentation.

Nous avons illustré sur la figure 5.10 les cartes de segmentations obtenues par le réseau mono-tâche (Fond/Ticket/Texte) et la dernière sortie du réseau multi-tâche avec les mêmes classes de segmentation. Ces images montrent que la localisation du texte et du ticket est meilleure avec le réseau multi-tâche.

3ème expérimentation : Évaluation des performances sur une architecture contenant trois sorties faiblement redondantes

Pour cette troisième expérimentation, on définit une architecture à quatre sorties, illustrée sur la figure 5.11. Chacune des sorties considérées réalise des tâches faiblement redondantes, en effet les tâches peuvent être complètement différentes :

— **Sortie 1** => Problème à deux classes : Bord des différentes zones ;
— **Sortie 2** => Problème à deux classes : Fond/Ticket ;
— **Sortie 3** => Problème à deux classes : Non Texte/Texte ;
— **Sortie 4** => Problème à quatre classes : Fond/Ticket/Logo/Texte.

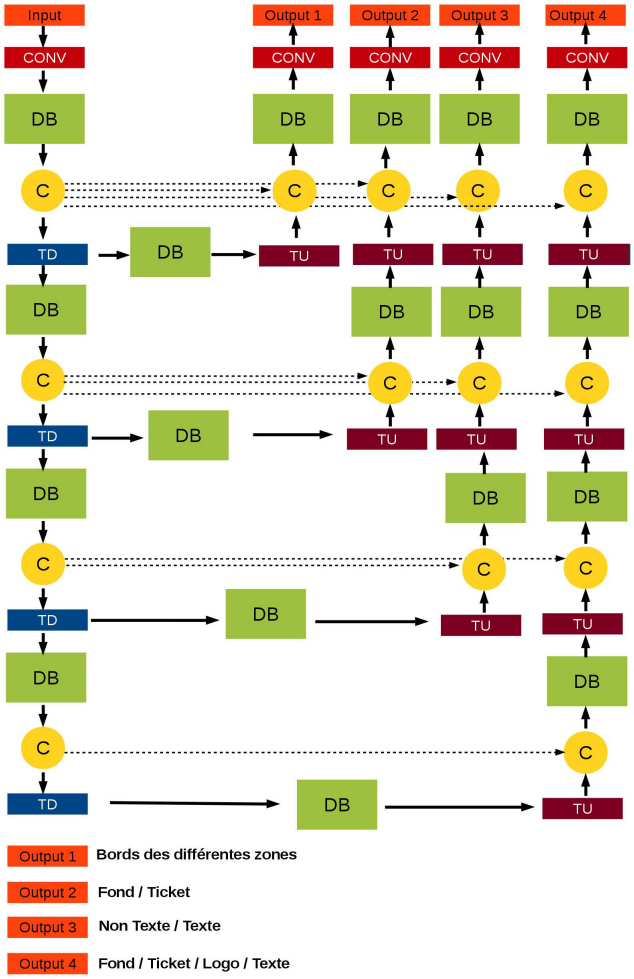


FIGURE 5.11. – Architecture multi-tâche basée sur le réseau FC-DenseNet, contenant quatre sorties faiblement redondantes.

Nombre de classes	Type Entraînement	Nb itérations	MeanIoU
Problème à 2 classes « Fond/Ticket »	Mono-tâche	600 000	0.59
	Multi-tâche	400 000	0.78
Problème à 3 classes « Fond/Ticket/Texte »	Mono-tâche	500 000	0.36
	Multi-tâche	400 000	0.65
Problème à 4 classes « Fond/Ticket/Logo/Texte »	Mono-tâche	600 000	0.32
	Multi-tâche	400 000	0.56

TABLE 5.6. – Résultats Mono-tâche et Multi-tâches sur problème à trois sorties redondantes. Tous les entraînements sont réalisés avec un BatchSize de 2 et un nombre d'epoch de 50.

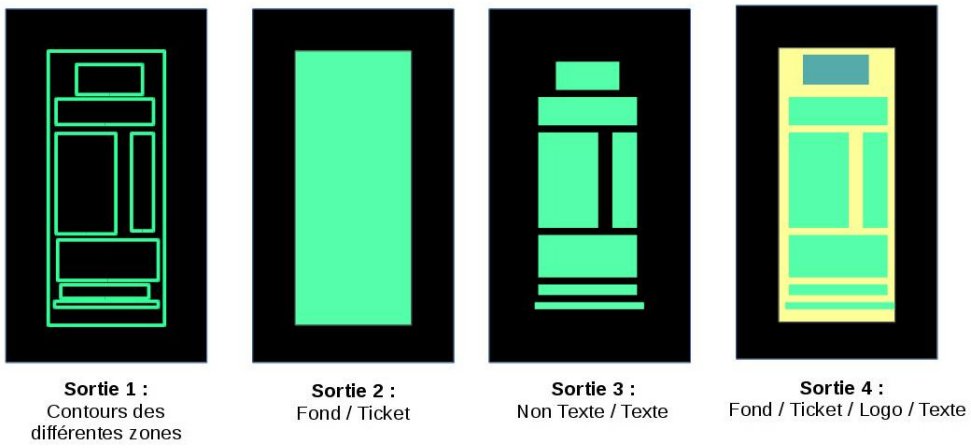


FIGURE 5.12. – Les cartes de segmentation des différentes sorties de la 3e expérimentation (voir paragraphe 5.3.3.b).

L'intérêt ici est d'évaluer si le caractère redondant des sorties de la deuxième expérimentation est responsable de l'amélioration des performances. La première sortie a pour but de localiser les contours de chaque blocs de texte, du ticket et du logo. La deuxième et la troisième sortie permettent d'extraire une entité (le « Ticket » ou le « Texte ») sur l'image, alors que la dernière sortie permet de dissocier le logo des autres blocs de texte. Les cartes de segmentation de ces quatre sorties sont illustrées sur la figure 5.12. Les résultats sont reportés sur le tableau 5.7. Au bout de 140 000 itérations les performances convergent.

Sur la figure 5.13, nous avons illustré les cartes de segmentation obtenues pour les problèmes à 4 classes, en utilisant d'une part le réseau entraîné en mono-tâche et d'autre part la dernière sortie du réseau entraîné en multi-tâche. On constate notamment que la localisation du logo est difficile mais qu'elle est largement améliorée par le multi-tâche.

Tâche de segmentation	Nb itérations	MeanIoU
Bordure	140 000	0.04
Fond/Ticket	140 000	0.74
Non Texte/Texte	140 000	0.69
Fond/Ticket/Logo/Texte	140 000	0.46

TABLE 5.7. – Résultats Multi-tâches sur problème à quatre sorties non redondantes.Tous les entraînements sont réalisés avec un BatchSize de 2 et un nombre d'epoch de 50.

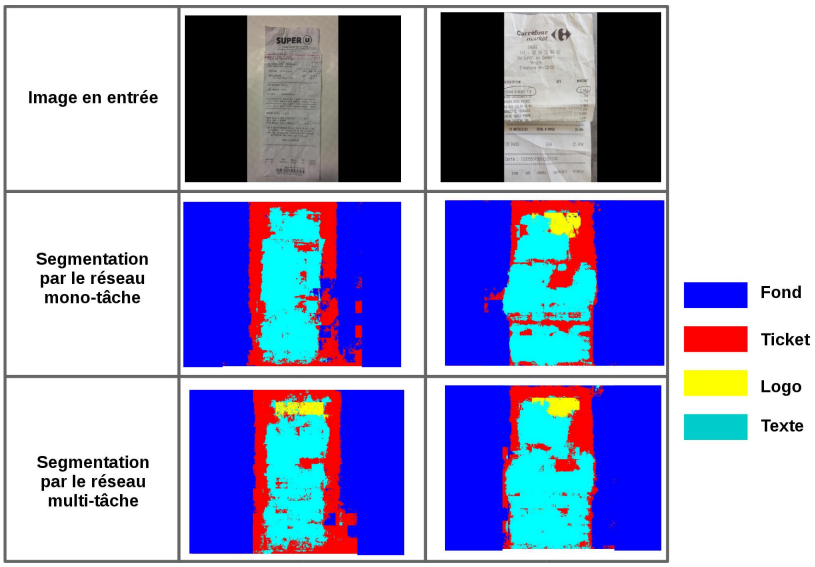


FIGURE 5.13. – Comparaison des cartes de segmentation pour le problème à 4 classes (Fond/Ticket/Logo/Texte) réalisées d'une part par le réseau mono-tâche et d'autre part par la dernière sortie du réseau multi-tâche.

On constate également, compte tenu des résultats décrits dans le tableau 5.7, que les performances atteintes sont toutes comparables à celles des expérimentations précédentes, mais avec un nombre d'itérations peu important.

Ces trois expérimentations ont permis de montrer que l'apprentissage multi-tâche permet d'améliorer les performances des différentes tâches de segmentation. Notamment, la simple introduction des bordures dans l'apprentissage multi-tâche permet d'améliorer nettement les résultats. Ce phénomène est clairement observé sur le problème, plus simple, d'une tâche de segmentation à deux classes : Fond/Ticket (cf. 5.5).

Cependant les performances atteintes restent améliorables pour permettre l'exploitation de ces réseaux dans le processus d'analyse de ticket. Des pistes de progrès résident dans l'enrichissement de la base d'apprentissage et dans une optimisation des pondérations des « loss » multi-tâches.
À ce jour, nous ne pouvons donc pas utiliser la catégorisation des blocs. En effet, elle n'est pas absolument nécessaire puisque la lecture du contenu peut compenser. Nous allons donc découper les blocs de texte par la simple détection des lignes blanches comme nous l'avons fait dans la mise en place du démonstrateur dans le chapitre 2 à partir de l'image du ticket rognée et redressée issue de la phase de pré-traitement.

5.4. La lecture du texte

La deuxième étape est la lecture des blocs ou des lignes de textes à l'aide d'un outil de reconnaissance optique de caractères. Puisque l'utilisation d'outils d'OCR disponibles sur le marché présente plusieurs inconvénients (voir section 5.2.2), il était important de pouvoir évaluer la faisabilité de créer un module de reconnaissance de caractères propres aux tickets de caisse, outil qui permettrait d'avoir une totale maîtrise sur ce processus.
En prenant en considération les derniers travaux sur la reconnaissance de caractères (voir chapitre 3 section 3.4), on constate que les approches basées sur les réseaux de neurones profonds ont permis d'atteindre des performances inédites. C'est pourquoi nous avons envisagé une approche de ce type.

Afin de pouvoir entraîner un tel système pour la classification de caractères spécifiques aux tickets de caisse, nous avons dû constituer une base d'apprentissage spécifique.

5.4.1. Base d'apprentissage

5.4.1.a. Constitution de la base des images de caractères

À partir des images à notre disposition, nous avons sélectionné une cinquantaine d'images de ticket issues d'enseignes différentes sur lesquelles nous avons segmenté les caractères de manière semi-automatique.

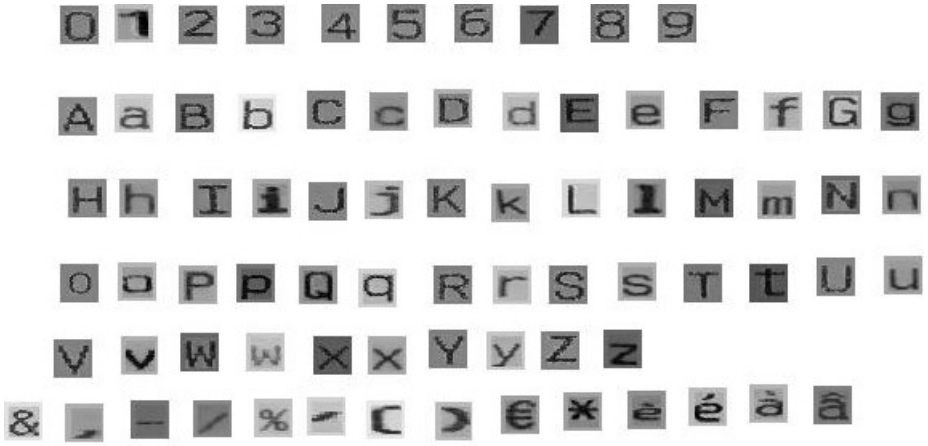


FIGURE 5.14. – Échantillon de la base de caractères construite.

Cette segmentation est faite en appliquant l'approche des projections présentée au chapitre 2. Pour que l'application de cette approche soit efficace, il y a un certain nombre de pré-requis à respecter, cette méthode étant très sensible à la qualité des images. C'est pourquoi la sélection des cinquante images est importante. Dans l'idéal elles doivent être relativement propres et avoir subi peu de déformations (pli, froissement, etc.). Toutes ces images ont été manuellement rognées pour éliminer l'arrière-plan afin de ne garder que le ticket dans l'image. C'est seulement à partir de ces images pré-traitées que l'approche présentée au chapitre 2 section 2.2.5 est applicable. Cette démarche comporte donc trois étapes : une binarisation, un filtrage éliminant le bruit de la binarisation et enfin la localisation des caractères par projections de l'intensité des pixels. Ces projections se font d'abord horizontalement pour isoler les différentes lignes, puis verticalement pour isoler les caractères. Une image contenant un unique caractère est générée et sauvegardée. Un tri manuel a été ensuite mis en place pour tout d'abord vérifier la qualité de la segmentation (caractères non coupés et présence d'un seul caractère dans l'image) et ensuite catégoriser chaque caractère.

Ainsi, nous avons obtenu 8 000 images de caractères appartenant à 76 classes différentes (en étant sensible à la casse) dont nous avons illustré un échantillon sur la figure 5.14.

5.4.1.b. Pré-traitement de la base de données

Afin d'optimiser l'apprentissage et la reconnaissance, les images de caractères collectées et classifiées sont pré-traitées.

Ajout d'une bordure :

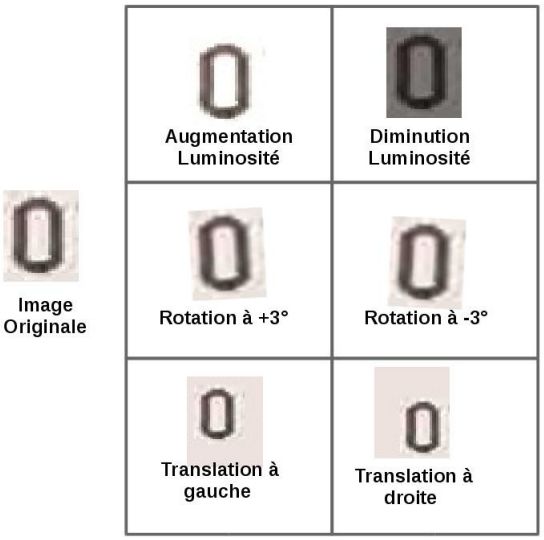


FIGURE 5.15. – Exemples de modifications appliquées à une image de caractère (changement de luminosité, rotation et translation.

La première étape est l'ajout d'un contour pour toutes les images de caractères. Cette étape permet d'assurer la lisibilité des caractères. En effet, nous avons constaté au chapitre 2 section 2.2.6 que tous les OCRs du marché sont sensibles à la taille de la boîte englobante de chaque caractère et qu'il était important d'étendre les boîtes englobantes pour que les contours extérieurs du caractère ne soient pas sur la frontière de la boîte englobante (« padding »), afin de ne pas masquer certains points d'intérêt. La couleur des pixels ainsi ajoutés correspond tout simplement à la valeur médiane des pixels de l'image, i.e la valeur du fond clair de l'imagette initiale.

Augmentation de la base de données :

Notre base de données étant réduite, nous procédons à une augmentation des données. L'entraînement d'un réseau de neurones profonds possédant un grand nombre de paramètres, il nécessite une quantité importante d'exemples pour obtenir de bonnes performances. Donc, pour obtenir plus de données, nous avons besoin de dupliquer les images initiales en introduisant des transformations illustrant la variabilité des données (voir figure 5.15). Les transformations que nous avons apportées sont des translations, des rotations (angle entre $-5°$ et $5°$) et des changements de luminosité. Le réseau de neurones considérera toutes ces images comme des images distinctes.
En effectuant un tirage aléatoire parmi ces transformations et en combinant systématiquement deux ou trois transformations, nous avons obtenu une base d'apprentissage d'environ 25 000 caractères.

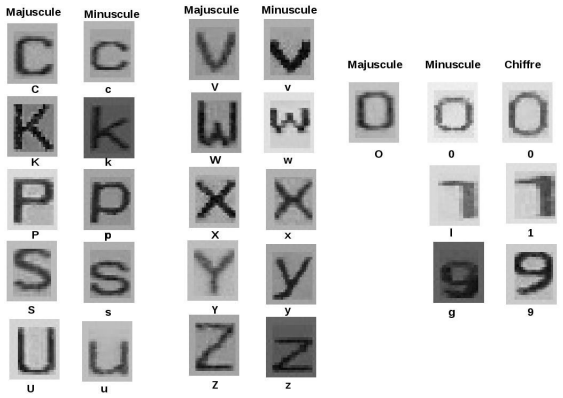


FIGURE 5.16. – Les différentes classes de caractères difficilement différentiables hors contexte et considérées comme appartenant à une classe unique pour la classification.

Finalisation de la base de données À partir de toutes ces images, nous avons défini trois bases différentes, avec environ 200 exemples par caractères :

— une première base sensible à la casse avec 76 caractères, en prenant en compte les caractères spéciaux (26 lettres minuscules, 26 lettres majuscules, 10 chiffres, 14 caractères spéciaux). La liste des caractères spéciaux considérés est répertoriée dans le tableau 5.8. Cette première base est notée « **B1** » ;

— une deuxième base insensible à la casse avec 50 caractères (26 lettres, 10 chiffres, 14 caractères spéciaux), notée « **B2** » ;

— une troisième base, sensible à la casse, mais dans laquelle les caractères peu différentiables hors contexte ont été rassemblés : comme par exemple le « O » majuscule , le « o » minuscule et le chiffre « 0 ». Les différentes classes que nous avons fusionnées sont représentées sur la figure 5.16, notée « **B3** ». Cette dernière base contient 63 classes.

5.4.2. Les Architectures utilisées

Parmi toutes les architectures récentes de l'état de l'art, celle qui correspond le mieux à notre situation est celle présentée par Max Jaderberg et al. dans [25]. Cette architecture permet de réaliser la détection et la reconnaissance de caractères, avec un taux de classification à 91% pour la reconnaissance sur le dataset de « ICDAR20113 ».

Nom du caractère spécial	Symboles	Images
Slash	« / »	
Pourcentage	« % »	
Tiret	« - »	
Parenthèse ouvrante	« (»	
Parenthèse fermante	«) »	
Symbole Euro	« € »	
Étoile	« * »	
Symbole « égal »	« = »	
Virgule	« , »	
Esperluette	« & »	
« e » accent aigu	« é »	
« e » accent grave	« è »	
« a » accent circonflexe	« â »	
« a » accent grave	« à »	

TABLE 5.8. – Les différents caractères spéciaux rencontrés sur l'échantillon des tickets de caisse sélectionnés.

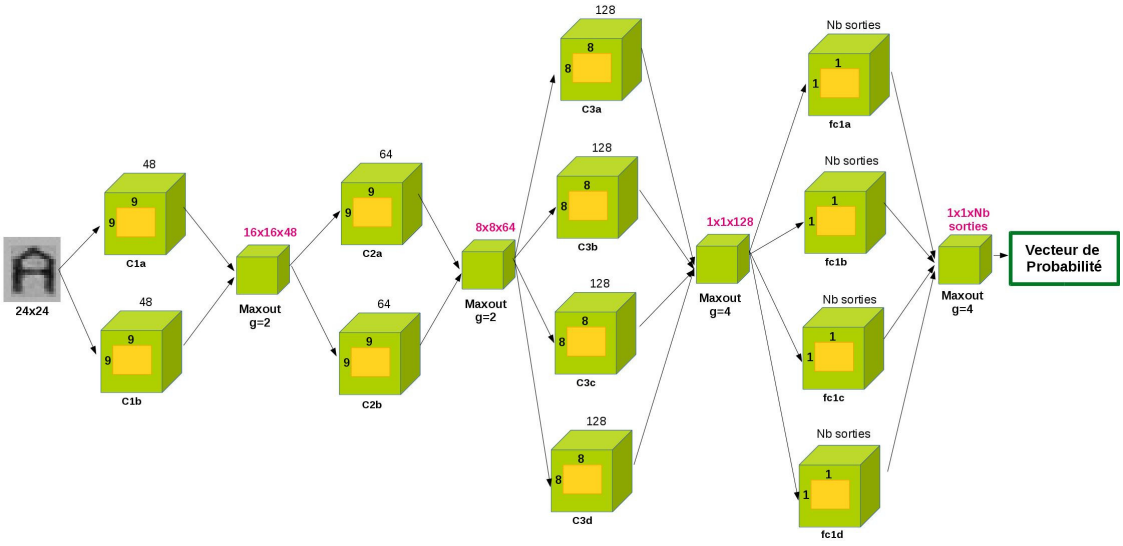


FIGURE 5.17. – Architecture pour la classification des caractères proposée par jaderberg et al. [25]. Description de l'architecture implémentée sur Caffe.

5.4.2.a. Description de l'architecture utilisée

Comme l'image en entrée est de petite taille, aucune opération de sous-échantillonnage spatial n'est effectuée. Le réseau est composé de quatre couches de convolutions et fait intervenir des opérations de « Maxout » (cf chapitre 3 section 3.2.4) entre chacune des couches de convolutions, comme illustré sur la figure 5.17. Sur les trois dernières couches, un dropout de 50% (cf chapitre 3 section 3.2.4) est appliqué afin d'éviter les risques de sur-apprentissage.

La première architecture que nous avons évaluée est exactement celle décrite par Jaderberg et al. dans [25], illustrée sur la figure 5.17. L'architecture [25] a été fidèlement appliquée sur nos données. Une variante sans dropout a également été testée.

5.4.3. Mise en œuvre et performances

5.4.3.a. Préparation de la base d'apprentissage

Afin que la base d'apprentissage corresponde aux deux architectures présentées ci-dessus, toutes les images ont été redimensionnées à la taille « 24x24 » qui correspond à la taille d'entrée des réseaux testés.
Pour les mêmes raisons évoquées pour la classification des logos (conditions d'éclairages, variétés des terminaux mobiles), les images sont converties en niveau de gris.

Concernant la répartition des données entre la phase d'apprentissage et la phase de

Taux de bonnes classifications			
	Architecture Originale		
Base	**Top1**	**Top3**	**Top5**
B1	0.73±0.02	0.91±0.01	0.95±0.02
B2	**0.93±0.02**	**0.95±0.01**	0.95±0.01
B3	0.76±0.01	0.91±0.02	**0.96±0.02**
	Architecture Sans Dropout		
B1	0.72±0.01	0.89±0.02	0.92±0.01
B2	0.92±0.01	0.94±0.01	0.94±0.01
B3	**0.88±0.03**	**0.92±0.01**	**0.95±0.01**

TABLE 5.9. – Performances des différentes architectures des réseaux de neurones entraînées pour la classification des caractères des tickets de caisses avec les bases **B1** (sensible à la casse), **B2** (insensible à la casse) et **B3** (sensible à la casse avec fusion des classes proches)

test, nous avons conservé la proportion suivante : 2/3 des données pour l'apprentissage et 1/3 des données pour le test. Afin de pouvoir mesurer les performances en validation croisée, nous avons répété systématiquement trois fois tous les apprentissages afin que toutes les images de la base de données aient été utilisées au moins une fois dans l'ensemble de tests et la moyenne des performances est reportée sur le tableau 5.9 .

5.4.3.b. Analyse des performances

Les performances obtenues pour la classification des caractères de tickets sont reportées dans le tableau 5.9. La différence de performance entre les deux architectures évaluées n'est pas nette et dépend de la base d'entraînement. La base « B1 », qui contient le plus grand nombre de classes (76 caractères), est celle qui obtient le moins bon taux de classification. La troisième base que nous avons proposée, base « B3 » dans laquelle nous avons fusionné les classes ressemblantes, permet d'améliorer la classification en Top5 et d'atteindre la performance maximale de 0.96. Le critère Top5 est intéressant car la connaissance des réponses proches permet de guider la correction sémantique qui suit la détection des caractères.

Il est difficile de comparer directement ces performances avec celles des OCR commerciaux car ces derniers sont optimisés pour la reconnaissance de mots et non de caractères, et utilisent parfois des dictionnaires intégrés. D'autre part, les performances obtenues avec les architectures proposées sont correctes, mais elles correspondent à des tests effectués sur une base de caractères sélectionnés. Dans une exploitation sur des images réelles, on peut s'attendre à une baisse de ces performances. Aussi, pour des raisons d'efficacité, la stratégie de l'entreprise a été de mettre en pause les développements d'un OCR spécifique et de privilégier, dans un premier temps, l'utilisation d'un OCR com-

mercial. Le choix s'est porté sur l'outil Google Vision qui a lu sans aucune erreur 94% des lignes segmentées sur 20 tickets correspondant à plus de 765 lignes (voir étude faite au chapitre 2).

5.5. Analyse Sémantique des textes lus

L'analyse sémantique des textes lus, i.e la compréhension du sens des textes, est la dernière étape du processus complet de l'analyse automatique des images de ticket de caisse. Cette étape va permettre l'exploitation des données extraites. L'objectif est d'identifier les informations du ticket de caisse : l'enseigne, la date d'achat, le nombre d'articles, le montant total, listes des produits achetés, etc.

C'est également à l'issue de cette étape qu'il va être possible de vérifier si toutes les informations recherchées ont bien été extraites, si elles sont cohérentes entre elles. Elle permet donc non seulement d'analyser les données présentes dans le contenu textuel du ticket, mais aussi de valider la qualité de l'analyse complète.

Cette dernière étape est en marge de ce travail de thèse, mais elle présente néanmoins un fort potentiel. Nous allons vous présenter ici les différentes actions mises en place afin :

— d'extraire les informations dites « de base », c'est-à-dire l'enseigne, la date d'achat, le nombre d'articles, le montant total etc ... ;
— d'identifier certains libellées courts.

Ces actions passent par l'enrichissement des bases de connaissances, afin d'affiner la recherche des mots-clés et la terminologie des enseignes.

5.5.1. Enrichissements des bases de connaissances

5.5.1.a. Création d'une plateforme

Une plateforme Web a été développée au sein de l'entreprise afin d'enrichir nos bases de connaissances sur l'analyse du texte des tickets de caisse. L'objectif est de pouvoir analyser manuellement/humainement toutes les images reçues dans la base de données de l'entreprise. Sur cette plateforme, l'extraction d'informations, que nous voulons automatiser, est faite manuellement. Les images ne contenant pas de ticket de caisse, ou contenant un ticket de caisse non conforme aux instructions (ticket illisible ou non complet, langue étrangère, etc.) sont écartées avec la justification qui correspond. Lorsque le ticket est analysable, toutes les informations pertinentes sont extraites, comme illustrées sur la figure 5.18. Toutes les informations relatives à une image sont ensuite sauvegardées dans une base de données.

Ce travail de collecte manuelle de données permet :

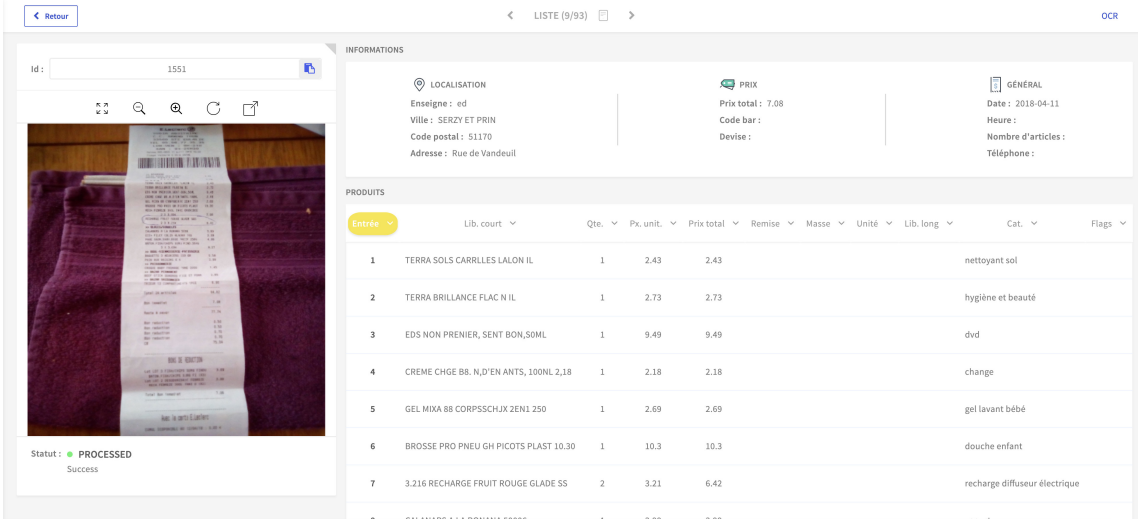


FIGURE 5.18. – Plateforme Web permettant la construction de la vérité terrain sur la base de ticket de caisse.

— de construire une vérité terrain sur un ensemble vaste et varié de tickets pour l'évaluation et la validation de l'analyse automatique ;
— d'enrichir les bases de connaissances (nom d'enseignes/magasins, numéro de téléphone, etc. ;
— d'appréhender au mieux les attentes et les difficultés de la lecture automatique d'un ticket de caisse pour un être humain, et par conséquent de trouver des solutions pour que l'analyse automatique s'effectue le mieux possible ;
— de rechercher des caractéristiques spécifiques des tickets de caisse, par enseignes ou par groupes d'enseignes ;
— d'identifier les ambiguïtés par exemple entre les noms de villes et de certains produits, la présence de différentes dates et de différents montants ;
— de comprendre et de rechercher des règles sur la construction des libellés courts.

5.5.1.b. Construction d'une base de produits

La deuxième source d'enrichissement des bases de connaissance est la construction d'une première base de données contenant plusieurs milliers de produits organisés selon les rayons des magasins. Elle permet d'associer chaque produit identifié manuellement ou automatiquement à un unique item de cette base. Cette base a été construite en récupérant les produits de certains sites des plus grandes enseignes de distribution alimentaire. Elle a été nettoyée manuellement pour éviter l'apparition de produit en doublon. Cependant cette organisation ne correspond pas à une organisation ontologique des produits et est difficile à maintenir. Le projet de l'entreprise est de réaliser une ontologie

propre des produits, ce qui nécessite une certaine maîtrise du domaine dont ne disposons pas actuellement, mais que nous acquérons au fur et à mesure des tickets analysés. Cette expertise est indispensable afin d'assurer que l'ontologie construite soit de qualité et la plus représentative possible.

5.5.2. Identification des libellés cours

L'identification des produits qualifiés au travers de libellés courts est un véritable challenge même pour les êtres humains. Cette tâche de traduction dépend des enseignes, voir des magasins. De plus, les règles définies par les distributeurs, s'ils en existent, ne sont pas connues. Par exemple, le libellé « pdm sdw ceral » correspondant au libellé long « pain de mie sandwich aux céréales » qui peut-être aisément déduit car plusieurs lettres (au moins 3) sont gardées pour définir les mots principaux de cet intitulé de produits. Mais d'autres exemples encore plus complexes, comme « lbvpav grmdepin&fr » associé au produit « le bon végétal pavé gourmand épinards & fromage » fait appel à des règles plus difficiles à identifier.

L'analyse manuelle de ces tickets nous a permis tout de même de confirmer quelques règles systématiquement respectées :

— La première lettre des mots principaux est toujours présente dans le libellé ;
— L'ordre des lettres dans les mots est respecté.

À partir de ces analyses manuelles nous avons également défini un dictionnaire de libellés courts fréquemment utilisés, comme par exemple « plt => poulet », « sch => sachet », « vqr => vache qui rit », « baf => boisson aux fruits », etc. Certains abrégés identiques peuvent être ambiguës et correspondre à plusieurs produits différents, comme « pdt » qui peut correspondre aux produits « Pomme de terre » où à la marque « Président » . Pour éviter toute confusion, nous avons également noté dans le dictionnaire le contexte permettant de lever l'ambiguïté comme les mots associés au libellé (par exemple la présence du mot « beurre » pour « pdt => President »).

La méthode utilisée consiste tout simplement à retrouver la signification de certains abrégés à l'aide du dictionnaire défini. Ensuite à l'aide d'un calcul de similarité via la méthode des N-grammes nous identifions l'« item » produit le plus proche dans la base de produits que nous avons construite.

Les performances d'identification de libellés courts sont bonnes. L'identification de libellés courts a été évaluée sur un ensemble de 3000 tickets, ce qui correspond à plus de 87 000 libellés courts. Le taux d'association correcte entre un libellé abrégé et le libellé du produit associé est de 88,7%. Parmi les erreurs constatées, nous avons identifié deux sources principales :

— des fautes d'orthographe sur les libellés courts qui peuvent survenir si l'OCR ne reconnaît pas correctement tous les caractères ou si l'étiquette a été créée à tort

Enseigne		Ville/Code Postal		Date et Heure		Montant Total		Nb Article	
Dém.	**Act.**	**Dém.**	**Act.**	**Dém.**	**Act.**	**Dém.**	**Act.**	**Dém.**	**Act.**
92%	97%	78%	80%	90%	90%	84%	88%	79%	80%

TABLE 5.10. – Performances de l'analyse Sémantique sur la base B1 lors de la construction du Démonstrateur (« Dém ») et les performances actuelles (« Act ») après les améliorations des algorithmes.

par un être humain ;
— présence d'ambiguïté dans la base de libellés produits.

5.5.3. Identification des informations de bases

Comme nous l'avons précisé au chapitre 2, les informations dites de base concernent l'enseigne, la date d'achat, le nombre d'articles, le montant total etc.

Afin d'améliorer les performances obtenues (et à défaut d'avoir une segmentation sémantique des blocs de texte), l'approche proposée est de délimiter les zones de recherche pour ces informations, en se focalisant soit sur l'entête du ticket, soit sur le bas du ticket. Nous avons vu chapitre 2 que la zone de libellés produits pouvait contenir des informations ambiguës. Pour cela, nous utilisons l'expression régulière décrite au chapitre 4 permettant de détecter une « ligne produit » afin de trouver la première et la dernière ligne du ticket correspondant à un produit acheté (cf chapitre 4). Ainsi il est possible de déterminer le début et la fin de la zone « libellé produit ». Les informations dites de base sont donc recherchées dans les zones extérieures à la zone des libellés produits (au dessus et en dessous).

Les performances obtenues ont pu être améliorées, comme on peut le voir dans le tableau 5.10, d'autant plus que la base de connaissance a été enrichie. La connaissance de l'enseigne en amont (grâce à la phase de pré-traitement décrite au chapitre 4) permet encore d'affiner la zone de recherche pour chacune des informations. En effet, pour chacune des grandes enseignes de distributions alimentaires rencontrées, nous avons décrit la structure type et la position précise de chacune des informations (entête, bas de ticket), ainsi que le contexte quand celui-ci est déterminant.

5.5.4. Conclusion

L'analyse sémantique est une étape critique de la lecture automatique du contenu d'un ticket de caisse. Cette analyse doit assurer l'extraction d'informations à partir de sources imparfaites (résultat de l'analyse d'image qui précède) et incomplètes (libellé abrégé). Parmi toutes les informations contenues dans un ticket de caisse, la liste des articles achetés est la plus délicate à extraire et est pourtant l'une des plus importantes du

point de vue de l'entreprise. Comme nous avons pu le voir, l'enrichissement des bases de connaissances ainsi que le développement d'algorithmes appropriés permettent d'atteindre des performances d'extraction intéressantes.

5.6. Synthèse et conclusion

Dans ce chapitre, nous nous sommes intéressée à l'analyse du contenu du ticket de caisse dans le but d'extraire les différents blocs de texte, de lire le contenu (OCR) et d'interpréter le résultat de cette lecture.

À partir des images « brutes » mais en tenant compte de toutes les informations issues de la phase de pré-traitement (présence d'un ticket, localisation grossière du ticket, orientation du ticket), l'approche proposée, basée sur un réseau de segmentation sémantique, permet de segmenter les différents blocs de texte en fonction de leur nature. Nous avons pu montrer que l'apprentissage multi-tâche permet d'améliorer les performances de segmentation.

Ensuite nous avons évalué la faisabilité d'un module de reconnaissance de caractères spécifiques aux tickets de caisse à partir d'une approche par apprentissage profond. Les performances de la méthode proposée sur la base de données que nous avons construites manuellement sont prometteuses, mais ne permettent pas encore de surpasser les performances des outils de reconnaissance optique de caractères sur le marché. À ce jour, nous continuons d'utiliser GoogleVision.

Enfin, en marge de ce travail, nous nous sommes intéressée à l'analyse sémantique des textes lus. L'objectif ici était d'apporter des solutions simples afin d'extraire au mieux les informations contenues dans le ticket, mais surtout d'appréhender les différentes difficultés de cette analyse et d'évaluer les différents moyens qui nous permettraient de les solutionner.

6. Conclusion générale et Perspectives

6.1. Conclusion

Ce travail de thèse intitulé a été réalisé dans le cadre d'une convention CIFRE avec la collaboration de l'entreprise AboutGoods Company à l'origine de ce projet. Nous avons exposé aux chapitres 1 et 2, les multiples intérêts et enjeux industriels d'un tel projet.

Ce travail a donc commencé par la réalisation d'un démonstrateur permettant d'évaluer la faisabilité d'un tel projet et de déterminer précisément les verrous scientifiques pour chacune des étapes. Après avoir examiné le cahier des charges exigeant du projet industriel ainsi que les performances des applications voisines, nous avons proposé des solutions simples pour chacune des étapes du processus qui sont au nombre de quatre :

— L'acquisition sur le terminal mobile ;
— La localisation du texte ;
— La reconnaissance du texte ;
— L'analyse sémantique.

À l'issue de l'évaluation de ce démonstrateur, nous avons proposé une chaîne de traitement pour la réalisation de ce lecteur automatique de ticket de caisse. Cette chaîne de traitement a eu pour objectif de répondre aux fortes contraintes industrielles, avec comme particularité la mise en place de différents algorithmes en compétition pour assurer la fiabilité de l'analyse. Cette chaîne de traitement comprend deux phases : une phase de pré-traitement (présentée au chapitre 4) et la phase d'analyse du contenu (présentée au chapitre 5).

Il est évident que la principale contribution de ce travail a été la mise en place d'un prototype de chaîne de traitement complète permettant la lecture automatique de tickets de caisse, puisqu'à notre connaissance aucune solution logicielle à ce jour ne propose de réaliser une lecture fiable et complète. La chaîne de traitement proposée est composée de différentes étapes. Différents outils sont utilisés à chacune des étapes pour répondre à un problème spécifique, la plupart des outils utilisés étant des outils existants. La contribution principale de cette thèse porte sur l'adaptation de ces outils dans le contexte spécifique de notre application afin de répondre aux objectifs fixés par l'entreprise.

Le système proposé sera, à terme, concrétisé sous forme d'une application ou d'un service pour les professionnels intéressés par l'analyse des contenus d'un ticket de caisse à

partir d'une photo (remboursement pour achat d'un produit, institut de sondage, étude de santé etc.), ou pour les particuliers en leur proposant un outil de gestion de leur budget. Cependant, il est clair que le système, dans son état actuel, est perfectible et qu'il existe des possibilités d'amélioration sur un certain nombre des étapes utilisées.

Dans la suite, nous résumons les contributions apportées à chaque étape de ce système de lecture de ticket de caisse. Nous proposons aussi des pistes d'amélioration pour donner une idée des axes possibles d'extension des travaux de recherche présentés dans cette thèse.

Détection et localisation du ticket

Nous avons présenté au chapitre 2 une solution permettant d'optimiser l'acquisition d'une image d'un ticket de caisse en temps réel sur un terminal mobile. Cette étape doit assurer une qualité d'image suffisante pour permettre l'analyse, mais sans trop contraindre l'utilisateur. Cependant, après une étude auprès des entreprises intéressées par ce système de lecture de ticket de caisse, il s'est avéré que l'accompagnement lors de l'acquisition, aussi souple soit-il, peut-être trop pénalisant et décourageant pour l'utilisateur. Il faut que la motivation de l'utilisateur à prendre en photo son ticket de caisse soit suffisamment importante pour qu'il accepte cette contrainte. Or il est difficile pour les entreprises de mesurer la motivation des utilisateurs de leurs applications. EN conséquence, elles préfèrent souvent ne pas prendre le risque de détourner ces utilisateurs et privilégient les acquisitions sans aucune contrainte.

Il a donc été important de proposer une alternative afin d'assurer la détection et la localisation du ticket dans l'image. Pour cela, nous nous sommes intéressée aux méthodes basées sur de l'apprentissage profond : le « Deep Learning ». À partir d'une base de données d'apprentissage, que nous avons manuellement collectée et annotée, nous avons entraîné un réseau de neurones en utilisant les méthodes de « transfert d'apprentissage ». Cette méthode nous permet d'entraîner un réseau de manière efficace malgré une faible quantité de données annotées. Ce réseau de neurones de classification est ensuite transformé en réseau de segmentation sémantique par patchs.

Ce réseau de neurones profond nous permet de répondre aux deux premières étapes du système :
— la détection d'un ticket dans l'image ;
— la localisation du ticket.

La même méthode est utilisée pour la classification et la localisation du logo de l'enseigne sur le ticket de caisse. En effet, nous avons entraîné un réseau de classification en utilisant la même méthode et nous avons également transformé ce réseau afin de pouvoir réaliser de la segmentation sémantique par patchs.

Fusion de méthodes d'analyse traditionnelle et d'apprentissage profond.

Afin de pouvoir prendre en compte les situations « difficiles » dues au fait que l'acquisition est laissée totalement libre, nous avons chercher à rendre le système développé robuste de manière à avoir un bonne fiabilité sur la lecture qui est effectuée. En prenant en compte la spécificité du document à étudier, le ticket de caisse, la fusion de plusieurs analyses différentes est proposée pour détecter, localiser le ticket de caisse et déterminer l'enseigne à travers le logo. Nous avons ainsi montré comment les méthodes traditionnelles et les approches basées sur les réseaux de neurones profonds sont complémentaires et que leur utilisation conjointe améliore les résultats finaux. Il est important de noter également que les différentes méthodes proposées visent à analyser différents aspects de l'image à étudier : l'information au niveau des pixels de l'image, ou de patch d'images, mais encore de l'information textuelle contenue dans l'image.

Nous avons proposé la fusion de plusieurs méthodes notamment sur les trois premières étapes du processus qui correspondent à la phase de pré-traitement (chapitre 4). Les algorithmes fusionnés sont réalisés en parallèle ou en cascade selon les étapes. Les outils de fusion utilisés sont généralement très simples et sont motivés par les intérêts industriels. En effet, il est crucial pour l'entreprise de ne perdre aucune information, aucune image intéressante et d'obtenir un degré de certitude suffisamment important pour chacune des données extraites.

Segmentation sémantique des zones du ticket.

La complexité de l'analyse sémantique du contenu textuel du contenu du ticket de caisse à partir d'une source de données incertaine nous a amenée à proposer une segmentation sémantique du ticket. En effet, l'analyse sémantique se base sur le texte extrait de l'image après différentes étapes d'analyse d'images et par un module de reconnaissance optique de caractères qui est rarement parfait. Et comme nous l'avons présenté au chapitre 2, certaines informations peuvent contenir de l'ambiguïté.

En se basant sur un réseau de neurones profond, nous avons proposé une segmentation des différentes zones du ticket de caisse (logo, entête, blocs produits etc.) en fonction de leur contenu sémantique. Nous avons montré que les performances de cette approche sont améliorables en proposant une architecture de réseau de neurones profond multi-tâche.

Cependant, à ce jour, cette segmentation sémantique nécessite d'être améliorée pour pouvoir être mise en place dans le processus d'analyse automatique. Nous avons proposé comme alternative une segmentation des blocs de texte dans un premier temps, puis des lignes dans un deuxième temps par projection de pixels. Nous avons également proposé

une solution permettant de délimiter les zones du ticket à l'aide de l'analyse sémantique des textes.

Reconnaissance Optique de Caractères.

Nous nous sommes intéressée à la réalisation d'un OCR spécifique aux ticket de caisse pour deux raisons. Tout d'abord l'intérêt est naturellement de tenter d'améliorer les performances de lecture, ensuite le second avantage, très important pour l'entreprise, est d'avoir une totale maîtrise et liberté d'utilisation de cet outil, ce qui n'est pas le cas avec les OCRs existants.
Malheureusement, malgré l'obtention de résultats encourageants, pour des raisons de dynamique de l'entreprise nous n'avons pas pu pousser plus le développement de cet OCR et nous nous sommes résignée à utiliser l'OCR GoogleVision qui est l'OCR le plus performant que nous avons testé.

Analyse sémantique.

En marge de ce travail, l'analyse sémantique des textes lus permettant la compréhension du sens des textes, constitue la dernière étape du processus complet de l'analyse automatique des images de ticket de caisse.

Nous avons notamment effectué de l'enrichissement des bases de connaissances sur le contenu de différents tickets de caisses, sur les différences et les points communs entre les enseignes et magasins, les produits, etc. Une étude manuelle sur des milliers de tickets de caisses nous a permis de mettre en évidence certaines caractéristiques et certaines règles dans la définition des libellés courts et même de définir des dictionnaires.

Grâce à l'analyse sémantique, une délimitation en trois zones du ticket est possible : l'entête, la liste des produits et le bas de ticket. Cette délimitation permet de réduire la zone de recherche de chaque information à extraire.

Tout ceci a permis d'améliorer les performances d'extraction des informations contenues dans le ticket de caisse.

6.2. Perspectives

Plusieurs perspectives sont envisageables.

Tout d'abord, l'accompagnement de l'utilisateur lors de l'acquisition est un point important qui permettra d'améliorer et de faciliter l'analyse qui suit. Aujourd'hui, les réseaux de neurones profonds sont de moins en moins lourds et peuvent être directement importés sur des terminaux mobiles qui sont eux de plus en plus puissants. D'autant plus, que des outils développés par les OS des terminaux mobiles facilitent l'intégration de ces réseaux dans des applications mobiles. De cette manière il sera possible de réaliser

la détection de la présence d'un ticket et la localisation du ticket dans l'image directement en local sur le mobile. Ainsi seules les images contenant un ticket de caisse, rogné et redressé seront envoyées sur le serveur pour être analysées.

Le deuxième axe d'amélioration est la segmentation sémantique des blocs de texte. Les performances obtenues dans cette thèse ne permettaient pas l'exploitation de cette méthode dans le processus complet. Cependant, il paraît évident que cette segmentation apporterait une réelle valeur ajoutée. Nous avons pu montrer que l'apprentissage multi-tâche permet une amélioration des performances, mais il reste encore plusieurs pistes à explorer autour de cette approche, en particulier l'utilisation de pondérations sur les fonctions de coût des différentes tâches. cette piste

Concernant la reconnaissance de l'enseigne, nous avons remarqué que la typographie, la police d'écriture permettaient de donner un indice très précis sur l'identification de l'enseigne. L'idée serait de développer un module de reconnaissance via un réseau de neurones profond permettant la classification des enseignes à partir de la typographie. Cette méthode autoriserait l'identification de l'enseigne par analyse d'image même en l'absence de logo.

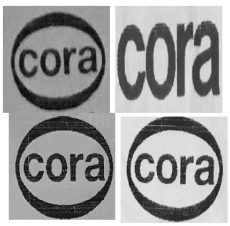
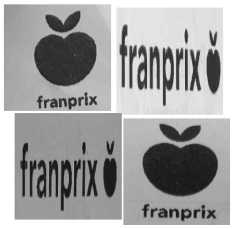
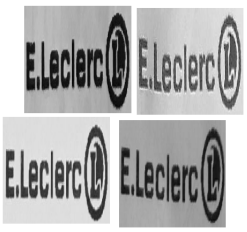
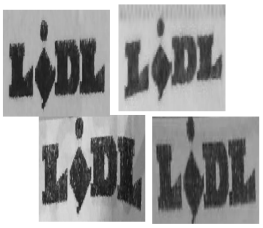
Nous soulignons que l'obtention d'un OCR dédié aux tickets de caisse est fondamentale pour une utilisation à grande échelle en entreprise. Les contraintes liées à l'utilisation d'un OCR du marché (coût, dépendance vis-à-vis de l'outil) sont trop lourdes sur le long terme ce qui explique la nécessité de s'approprier de telles technologies.

Toujours concernant l'analyse d'image, certaines situations que nous avons rencontrées ne sont pas analysables par le système de lecture automatique que nous avons développé. C'est le cas des tickets très longs ou des images contenant plusieurs tickets. Pour les tickets très longs, qui rendent difficile la phase d'acquisition, certains utilisateurs parviennent à prendre une unique photo du ticket, mais celle-ci est difficilement lisible. D'autres utilisateurs prennent le ticket en plusieurs fois, ce qui complique l'analyse d'image car il faut pouvoir reconstituer le ticket à partir des différentes photos. Une solution serait de pouvoir guider l'utilisateur lors de l'acquisition d'un ticket long en mode « panorama ». Pour les images contenant plusieurs tickets différents, notre système ne valide pas ce type d'image, notamment à cause de la présence d'incohérences (différentes enseignes, différentes villes, etc.). La possibilité de détecter la présence de plusieurs tickets sur une même image permettrait de pouvoir les exploiter.

Finalement, l'analyse sémantique des textes lus nécessite d'être mieux exploitée, en commençant par la définition et la maintenance d'une ontologie des produits alimentaires ce qui permettrait d'améliorer l'identification des produits achetés. Afin d'automatiser la conversion des libellés courts en libellés longs, il serait également intéressant d'étudier l'apport et les performances d'un algorithme, basé sur les n-grammes et s'appuyant sur cette ontologie et des différentes bases de connaissances (dictionnaire, lexique, etc.).

7. Annexe 1

Dans cet annexe, nous présentons un échantillon des différents exemples pour chaque logo de la base d'apprentissage que nous avons manuellement annoté (cf Chapitre 4)

Carrefour	Carrefour Market	Carrefour City	Carrefour Contact
			
Casino	**Cora**	**Franprix**	**Géant**
			
Grand Frais	**Hyper U**	**Intermarché**	**E.Leclerc**
			
Lidl	**Match**	**Migros**	**Monoprix**
			
Super U	**U Express**		
			

Bibliographie

[1] D. Doermann, "The indexing and retrieval of document images : A survey.," Computer Vision and Image Understanding, 1998.

[2] O. Frieder D. Grossman G. Agam, S. Argamon and D. Lewis, "Content-based document image retrieval in complex document collections.," Proc, 2007.

[3] Simone Marinai, Simone Marinai, and Hiromichi Fujisawa, Machine Learning in Document Analysis and Recognition, Springer Publishing Company, Incorporated, 1st edition, 2008.

[4] JP.Domenger O.Augereau, N.Journet, "Semi-structured document image matching and recognition," Document Recognition and Retrieval, February 2013.

[5] D. Gaceb, Contributions au tri automatique de documents et de courrier d'entreprises., Ph.D. thesis, Institut National de Sciences Appliquées de Lyon, 2009.

[6] R. Jayadevan, S. R. Kolhe, P. M. Patil, and U. Pal, "Automatic processing of handwritten bank cheque images : A survey," Int. J. Doc. Anal. Recognit., vol. 15, no. 4, pp. 267–296, Dec. 2012.

[7] H. Hamza, Y. Belaid, A. Belaid, and B. B. Chaudhuri, "Incremental classification of invoice documents," in 2008 19th International Conference on Pattern Recognition, Dec 2008, pp. 1–4.

[8] E. Sorio, A. Bartoli, G. Davanzo, and E. Medvet, "A domain knowledge-based approach for automatic correction of printed invoices," in International Conference on Information Society (i-Society 2012), June 2012, pp. 151–155.

[9] M. Diligenti, P. Frasconi, and M. Gori, "Hidden tree markov models for document image classification," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25, no. 4, pp. 519–523, April 2003.

[10] Bill Janssen, Eric Saund, Eric A. Bier, Patricia Wall, and Mary Ann Sprague, "Receipts2go : the big world of small documents," in ACM Symposium on Document Engineering, 2012.

[11] Roland Szabo, "A novel machine learning based approach for retrieving information from receipt images," Mis en ligne le 15 avril 2014.

[12] S. Garimella, "Identification of receipts in a multi-receipt image using spectral clustering," International Journal of Computer Applications, vol. 155, No. 2, Dec. 2016.

[13] Ozhiganov Ivan, "Applying ocr technology for receipt recognition," Mis en ligne le 07 avril 2016.

[14] M. Jones P. Viola, "Rapid object detection using a boosted cascade of simple features," CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2001.

[15] A. Suponenkovs, A. Sisojevs, G. Mosāns, J. Kampars, K. Pinka, J. Grabis, A. Locmelis, and R. Taranovs, "Application of image recognition and machine learning technologies for payment data processing," in 5th IEEE Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE), November 2017.

[16] N.Otsu, "A threshold selection method from grey scale histogram," IEEE Trans. on Syst. Man and Cyber, 1979.

[17] E. Lecolinet R.G. Casey, "A survey of methods and strategies in character segmentation," IEEE Transactions on Pattern Analysis and Machine Intelligence, July 1996.

[18] Yi Lu, "Machine printed character segmentation —; an overview," IEEE Transactions on Pattern Analysis and Machine Intelligence, January 1995.

[19] D. Lee R. Smith, D. Antonova, "Adapting the tesseract open source ocr engine for multilingual ocr," ACM International Workshop on Multilingual OCR, July 2009.

[20] "Ocr, pdf, text scanning software and solutions - abbyy," .

[21] "Cloud vision api powerful image analysis," .

[22] S. Delisle I. Biskri, "Les n-grams de caractères pour l'aide à l'extraction de connaissances dans des bdd textuelles multilingues," TALN, 2001.

[23] R. William Soukoreff and I. Scott MacKenzie, "Measuring errors in text entry tasks : An application of the levenshtein string distance statistic," pp. 319–320, 2001.

[24] Yu Qiao Chen Change Loy Xiaoou Tang Pan He, Weilin Huang, "Reading scene text in deep convolutional sequences," arXiv :1506.04395, June 2015.

[25] Jaderberg M., Vedaldi A., and Zisserman A., "Deep features for text spotting," ECCV, 2014.

[26] Andrea Vedaldi Andrew Zisserman Max Jaderberg, Karen Simonyan, "Reading text in the wild with convolutional neural networks," arXiv :1412.1842, December 2014.

[27] Yali Li, Shengjin Wang, Qi Tian, and Xiaoqing Ding, "Feature representation for statistical-learning-based object detection : A review," Pattern Recognition, vol. 48, no. 11, pp. 3542 – 3559, 2015.

[28] Nicholas Ayache and Olivier Faugeras, "HYPER : a new approach for the recognition and positioning of 2D objects," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 8, no. 1, pp. 44–54, 1986.

[29] G. Csurka, C. R. Dance, L. Fan, J. Willarnowski, and C. baray, "Visual categorization with bags of keypoints.," ECCV, 2004.

[30] D.E Rumelhart, G. E. Hinton, and R.J.Williams, "Learning internal representations by error propagation.," Explorations in the Microstructure of Cognition, 1986.

[31] G. E. Hinton A.Krizhevsky, I.Sutskever, "Imagenet classification with deep convolutional neural networks," NIPS, 2012.

[32] V. Nair and G.E. Hinton, "Evaluation of pooling operations in convolution architectures for object recognition.," ICANN, 2010.

[33] D. Scherer, A. Muller, and S.Behnke, "Rectified linear units improve restricted boltzmann machines.," International Conference on Machine Learning, 2010.

[34] Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio, "Maxout networks," pp. III–1319–III–1327, 2013.

[35] J. Deng, A. Berg, S. Satheesh, H. Su, A. Khosla, and L. Fei-Fei, "http ://www.image-net.org/challenges/lsvrc/2012/," ILSVRC-2012, 2012.

[36] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," ICLR, 2015.

[37] Y.Jia P.Sermanet S.Reed D.Anguelov D.Erhan V.Vanhoucke A.Rabinovich C.Szegedy, W.Liu, "Going deeper with convolutions," CVPR, 2014.

[38] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," CVPR, 2016.

[39] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks," arXiv :1411.1792, 2014.

[40] J. Donahue, Y Jia, O. Vinyals, J. Hoffman, J. Zhang, E .Tzeng, and T. Darrell, "Decaf :a deep convolutional activation feature for generic visual recognition.," Technical Report, Arxiv :1310.1531, 2013.

[41] P.Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat : Integrated recognition, localization and detection using convolutional networks.," International Conference on Learning Representation, 2014.

[42] M. D. Zeiler. and R. Fergus, "Visualizing and understanding convolutional networks," Techincal Report, Arxiv 1311.2901, 2013.

[43] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," Techincal Report, 2013.

[44] X. Zhang andd M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat : Integrated recognition, localization and detection using convolutionl networks.," Proceedings of international Conference on Learning Representation (ICLR), 2014.

[45] C. Szegedy, A. Toshev, D. Erhan, and G. Inc, "Deep nueral networks for object detection.," Advances in Neural Information Processing Systems (NIPS), 2014.

[46] R. Girshick, J. Donahue, T. Darell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation.," CVPR, 2014.

[47] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders, "Selective search for object recognition.," IJCV, 2013.

[48] Ross Girshic, "Fast r-cnn," arXiv :1504.08083, 2015.

[49] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun, "R-FCN : object detection via region-based fully convolutional networks," CoRR, vol. abs/1605.06409, 2016.

[50] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg, "SSD : single shot multibox detector," CoRR, vol. abs/1512.02325, 2015.

[51] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick, "Mask R-CNN," CoRR, vol. abs/1703.06870, 2017.

[52] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun, "Faster R-CNN : towards real-time object detection with region proposal networks," CoRR, vol. abs/1506.01497, 2015.

[53] ," .

[54] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition, Washington, DC, USA, 2011, CVPR '11, pp. 1297–1304, IEEE Computer Society.

[55] Jonathan Long, Evan Shelhamer, and Trevor Darrell, "Fully convolutional networks for semantic segmentation," CoRR, vol. abs/1411.4038, 2014.

[56] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla, "Segnet : A deep convolutional encoder-decoder architecture for image segmentation," CoRR, vol. abs/1511.00561, 2015.

[57] Eli David and Nathan Netanyahu, "Deeppainter : Painter classification using deep convolutional autoencoders," 09 2016, pp. 20–28.

[58] Fisher Yu and Vladlen Koltun, "Multi-scale context aggregation by dilated convolutions," CoRR, vol. abs/1511.07122, 2015.

[59] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille, "Deeplab : Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," CoRR, vol. abs/1606.00915, 2016.

[60] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello, "Enet : A deep neural network architecture for real-time semantic segmentation," CoRR, vol. abs/1606.02147, 2016.

[61] Simon Jégou, Michal Drozdzal, David Vázquez, Adriana Romero, and Yoshua Bengio, "The one hundred layers tiramisu : Fully convolutional densenets for semantic segmentation," CoRR, vol. abs/1611.09326, 2016.

[62] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger, "Densely connected convolutional networks," CoRR, vol. abs/1608.06993, 2016.

[63] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-net : Convolutional networks for biomedical image segmentation," CoRR, vol. abs/1505.04597, 2015.

[64] Ronan Collobert and Jason Weston, "A unified architecture for natural language processing : Deep neural networks with multitask learning," pp. 160–167, 2008.

[65] Z. Kuang, Z. Li, T. Zhao, and J. Fan, "Deep multi-task learning for large-scale image classification," pp. 310–317, April 2017.

[66] Li Deng, Geoffrey Hinton, and Brian Kingsbury, "New types of deep neural network learning for speech recognition and related applications : An overview," 2013.

[67] Sebastian Ruder, "An overview of multi-task learning in deep neural networks," CoRR, vol. abs/1706.05098, 2017.

[68] Mingsheng Long and Jianmin Wang, "Learning multiple tasks with deep relationship networks," CoRR, vol. abs/1506.02117, 2015.

[69] Alex Kendall, Yarin Gal, and Roberto Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," CoRR, vol. abs/1705.07115, 2017.

[70] Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher, "A joint many-task model : Growing a neural network for multiple NLP tasks," CoRR, vol. abs/1611.01587, 2016.

[71] Boris Mansencal, Jenny Benois-Pineau, Hervé Bredin, Alexandre Benoit, Nicolas Voiron, Patrick Lambert, and Georges Quénot, "IRIM at TRECVID 2016 : Instance Search," Nov. 2016.

[72] Hang Su, Xiatian Zhu, and Shaogang Gong, "Deep learning logo detection with data expansion by synthesising context," pp. 530–539, 2017.

[73] Hang Su, Shaogang Gong, and Xiatian Zhu, "Weblogo-2m : Scalable logo detection by deep learning from the web," Oct 2017.

[74] Simone Bianco, Marco Buzzelli, Davide Mazzini, and Raimondo Schettini, "Deep learning for logo recognition," Neurocomputing, vol. 245, pp. 23–30, 2017.

[75] A. Bagdanov, L. Ballan, M. Bertini, and A. Del Bimbo, "Trademark matching and retrieval in sports video databases.," In Proceedings of the international workshop on Workshop on multimedia information retrieval, 2007.

[76] S. Romberg and R. Lienhart, "Bundle min-hashing for logo recognition.," In Proceedings of the 3rd ACM Conference on International Conference on multimedia retrieval, 2013.

[77] S. Romberg, R. Lienhart, and R. Van Zwol, "Scalable logo recognition in real-word images.," In Proceedings of the 1st ACM International Conference on multimedia retrieval, 2011.

[78] R. Boia, C. Florea, L.Florea, and R.Dogaru, "Logo localization and recognition in natural images using homographic class graphs.," Machine Vision and Application, February 2016.

[79] R. Boia, C. Florea, and L.Florea, "Elliptical asift agglomeration in class prototype for logo detection.," BMVC, 2015.

[80] Christian Eggert, Anton Winschel, and Rainer Lienhart, "On the benefit of synthetic data for company logo detection.," Proceedings of the 23rd ACM international conference on Multimedia, October 2015.

[81] S. Bianco, M. Buzzelli, D. Mazzini, and R. Schettini, "Logo recognition using cnn features," in International Conference on Image Analysis and Processing. Springer, 2015, pp. 438–448.

[82] Forrest N. Iandola, A.Shen, P.Gao, and K.Keutzer, "Deeplogo : Hitting logo recognition with the deep neural network hammer," arXiv :1510.02131, October 2015.

[83] Gonçalo Oliveira, Xavier Frazão, André Pimentel, and Bernardete Ribeiro, "Automatic graphic logo detection via fast region-based convolutional networks," CoRR, vol. abs/1604.06083, 2016.

[84] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, "Return of the devil in the details : Delving deep into convolutional nets.," arXiv preprint arXiv :1405.3531, 2014.

[85] S. C. Hoi, X. Wu, H. Liu, Y. Wu, H. Wang, H. Xu, and Q. Wu, "Logo-net : Large-scale deep logo detection and brand recognition with deep region-based convolutional networks.," arXiv :1511.02462, 2015.

[86] Anthimopoulos M., B. Gatos, and Pratikakis, "Detection of artificial and scene text in images and video frames," Pattern Anal Applic, 2013.

[87] Huizhong Chen, Sam S. Tsai, Georg Schroth, David M. Chen, Radek Grzeszczuk, and Bernd Girod, "Robust text detection in natural images with edge-enhanced maximally stable extremal regions," sept 2011.

[88] Xu-Cheng Yin, Xuwang Yin, Kaizhu Huang, and Hong-Wei Hao, "Robust text detection in natural scene images," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 36, no. 5, pp. 970–983, May 2014.

[89] Anand Mishra, Karteek Alahari, and C.V. Jawahar, "Scene Text Recognition using Higher Order Language Priors," Sept. 2012.

[90] Tatiana Novikova, Olga Barinova, Pushmeet Kohli, and Victor Lempitsky, "Large-lexicon attribute-consistent text recognition in natural images," in ECCV, 2012, pp. 752–765.

[91] J.Matas L.Neumann, "Scene text localization and recognition with oriented stroke detection.," CVPR, December 2013.

[92] Jerod J. Wainman, Zacary Butler, Dugan Knoll, and Jacqueline Feild, "Toward integrated scene text reading," EEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, February 2014.

[93] Alessandro Bissacco, Mark Cummins, Yuval Netzer, and Hartmut Neven, "Photoocr : Reading text in uncontrolled conditions," pp. 785–792, 2013.

[94] Ouais Alsharif and Joelle Pineau, "End-to-end text recognition with hybrid hmm maxout models.," CoRR, vol. abs/1310.1811, 2013.

[95] D.Karatzas L.Gomez, "Multi-script text extraction from natural scenes. computer vision center ipc .," Computer Vision Center IPC, 2013.

[96] J.Matas L.Neumann, "Real-time scene text localization and recognition.," CVPR, June 2013.

[97] L. Neumann and J. Matas, "Text localization in real-world images using efficiently pruned exhaustive search," pp. 687–691, Sept 2011.

[98] Kai Wang, Boris Babenko, and Serge Belongie, "End-to-end scene text recognition," pp. 1457–1464, 2011.

[99] Chitrakala Gopalan and D. Manjula, "Sliding window approach based text binarisation from complex textual images," (IJCSE) International Journal on Computer Science and Engineering, November 2010.

[100] A. Nicolaou and B. Gatos, "Handwritten text line segmentation by shredding text into its lines," pp. 626–630, July 2009.

[101] Vladimir Shapiro, Georgi Gluhchev, and Vassil Sgurev, "Handwritten document image segmentation and analysis," Pattern Recognition Letters, vol. 14, no. 1, pp. 71 – 78, 1993.

[102] Xiangyu Zhu, Yingying Jiang, Shuli Yang, Xiaobing Wang, Wei Li, Pei Fu, Hua Wang, and Zhenbo Luo, "Deep residual text detection network for scene text," CoRR, vol. abs/1711.04147, 2017.

[103] Fan Jiang, Zhihui Hao, and Xinran Liu, "Deep scene text detection with connected component proposals," CoRR, vol. abs/1708.05133, 2017.

[104] Zhuoyao Zhong, Lianwen Jin, Shuye Zhang, and Ziyong Feng, "Deeptext : A unified framework for text proposal generation and text detection in natural images," CoRR, vol. abs/1605.07314, 2016.

[105] Zheng Zhang, Chengquan Zhang, Wei Shen, Cong Yao, Wenyu Liu, and Xiang Bai, "Multi-oriented text detection with fully convolutional networks," CoRR, vol. abs/1604.04018, 2016.

[106] Minghui Liao, Baoguang Shi, Xiang Bai, Xinggang Wang, and Wenyu Liu, "Textboxes : A fast text detector with a single deep neural network," CoRR, vol. abs/1611.06779, 2016.

[107] Baoguang Shi, Xiang Bai, and Serge J. Belongie, "Detecting oriented text in natural images by linking segments," CoRR, vol. abs/1703.06520, 2017.

[108] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng, "End-to-end text recognition with convolutional neural networks," pp. 3304–3308, Nov 2012.

[109] C. Yao, X. Bai, B. Shi, and W. Liu, "Strokelets : A learned multi-scale representation for scene text recognition," pp. 4042–4049, June 2014.

[110] Ouais Alsharif and Joelle Pineau, "End-to-end text recognition with hybrid HMM maxout models," CoRR, vol. abs/1310.1811, 2013.

[111] A. Bissacco, M. Cummins, Y. Netzer, and H. Neven, "Photoocr : Reading text in uncontrolled conditions," pp. 785–792, Dec 2013.

[112] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng, "End-to-end text recognition with convolutional neural networks," pp. 3304–3308, Nov 2012.

[113] Anand Mishra, Karteek Alahari, and C V. Jawahar, "Scene text recognition using higher order language priors," 09 2012.

[114] Tatiana Novikova, Olga Barinova, Pushmeet Kohli, and Victor Lempitsky, "Large-lexicon attribute-consistent text recognition in natural images," October 2012.

[115] Vibhor Goel, Anand Mishra, Karteek Alahari, and C. V. Jawahar, "Whole is Greater than Sum of Parts : Recognizing Scene Text Words," Aug. 2013.

[116] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, "Reading text in the wild with convolutional neural networks," CoRR, vol. abs/1412.1842, 2014.

[117] Bastien Moysset, Christopher Kermorvant, and Christian Wolf, "Full-page text recognition : Learning where to start and when to stop," CoRR, vol. abs/1704.08628, 2017.

[118] M. E. Roberts B. M. Stewart A. Storer Dustin Tingley C. Lucas, R.A. Nielsen, "Computer-assisted text analysis for comparative politics," Political Analysis (2015) 23 (2) : 254-277, February 2015.

[119] Richard C. Wang Estevam R. Hruschka Tom M. Mitchell A.Carlson, J.Betteridge, "Coupled semi-supervised learning for information extraction," WSDM '10 Proceedings of the third ACM international conference on Web search and data mining, Pages 101-110, Fevruary 2010.

[120] Christophe Roche, "Le terme et le concept : fondements d'une ontoterminologie," CoRR, vol. abs/0801.1275, 2008.

[121] Sorgel D. Aussenac-Gilles N., "Text analysis for ontology and terminology engineering," Applied Ontology, Volume 1 Issue 1, Pages 35-46, January 2005.

[122] M.Filipe Santos F. Mota Pinto, A.Marques, "Ontology-supported database marketing," Journal of Database Marketing Customer Strategy Management, March 2009.

[123] William J. Frawley, Gregory Piatetsky-Shapiro, and Christopher J. Matheus, "Knowledge discovery in databases : An overview," AI Mag., vol. 13, no. 3, pp. 57–70, Sept. 1992.

[124] Lisa Di Jorio, Lylia Abrouk, Céline Fiot, Maguelonne Teisseire, and Danièle Hérin, "Enrichissement d'ontologie basé sur les motifs séquentiels," July 2007.

[125] H.Stuckenschmidt G.Schuster, "Building shared ontologies for terminology integration," Workshop on "Ontologies", Affiliated with KI-2001, September 2001.

[126] Kunyanuth Kularbphettong, "Enrichment ontology instance by using data mining techniques," pp. 150–155, 2018.

[127] Claudia d'Amato, Steffen Staab, Andrea G. B. Tettamanzi, Tran Duc Minh, and Fabien Gandon, "Ontology enrichment by discovering multi-relational association rules from ontological knowledge bases," pp. 333–338, 2016.

[128] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean, "Distributed representations of words and phrases and their compositionality," pp. 3111–3119, 2013.

[129] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Marc Aurelio Ranzato, and Tomas Mikolov, "Devise : A deep visual-semantic embedding model," pp. 2121–2129, 2013.

[130] Jianfeng Gao, Xiaodong He, Wen-tau Yih, and Li Deng, "Learning semantic representations for the phrase translation model," CoRR, vol. abs/1312.0482, 2013.

[131] Xiang Zhang and Yann LeCun, "Text understanding from scratch," CoRR, vol. abs/1502.01710, 2015.

[132] A.Benoit P.Lambert R.Raoui-Outach, C.Million-Rousseau, "Lecture automatique d'un ticket de caisse par vision embarquee sur un telephone mobile.," RFIA, July 2016.

Table des figures

Liste des tableaux