THÈSE POUR OBTENIR LE GRADE DE DOCTEUR DE L'UNIVERSITE DE MONTPELLIER

En Informatique

École doctorale : Information, Structures, Systèmes

Unité de recherche LIRMM

Localisation d'objets urbains à partir de sources multiples dont des images aériennes

Présentée par Lionel Pibre

Le 30 novembre 2018

Sous la direction de Marc Chaumont
Gérard Subsol et Dino Ienco

Devant le jury composé de

Jenny Benois-Pineau, Pr, Univ. Bordeaux /CNRS/IPB-ENSEIRB	Rapportrice
Sébastien Lefèvre, Pr, Univ. de Bretagne Sud/IRISA IUT Vannes	Rapporteur
Josiane Zerubia, DR, INRIA Sophia-Antipolis Méditerranée	Examinatrice
Georges Quénot, DR, CNRS/LIG Grenoble	Examinateur
Jérôme Azé, Pr, Univ. Montpellier/LIRMM	Président du jury
Laurent Deruelle, MCF, LRA - Berger-Levrault	Invité
Gérard Subsol, CR, CNRS/LIRMM	Co-encadrant
Dino Ienco, CR, IRSTEA, UMR TETIS	Co-encadrant
Marc Chaumont, MCF, Univ. Nîmes/LIRMM	Directeur

UNIVERSITÉ DE MONTPELLIER

Remerciements

Tout d'abord, je tiens à remercier Jenny BENOIS-PINEAU et Sébastien LEFÈVRE qui ont accepté de relire ce manuscrit et d'être rapporteurs de ma thèse. Leur lecture attentive et leurs remarques précises m'ont permis d'améliorer la version finale de ce manuscrit.

Je remercie également Josianne ZÉRUBIA et George QUÉNOT d'avoir été examinateurs de ma thèse et pour leurs propositions d'améliorations du manuscrit très pertinentes. Enfin, je remercie Jérôme AZÉ d'avoir présidé mon jury de thèse.

Je voudrais remercier mon directeur de thèse Marc CHAUMONT et mes encadrants, Gérard SUBSOL et Dino IENCO qui m'ont encadré et dirigé pendant la durée de ma thèse. Merci pour leur aide, leurs conseils et leur disponibilité qui ont été pour moi des points essentiels pour comprendre et m'adapter aux enjeux de la recherche. Leurs nombreuses relectures et corrections de mes travaux, en particulier ceux de cette thèse, ont été très importantes pour leurs qualités finales. Travailler sous leur direction a été très agréable et pour cela merci.

Je tiens aussi remercie Mustapha DERRAS et Laurent DERUELLE de la société Berger-Levrault de m'avoir accordé leur confiance et de m'avoir permis d'effectuer ma thèse dans de bonnes conditions. De plus, je les remercie de m'avoir permis de participer à des conférences sans aucune difficulté.

Merci à Philippe BORIANNE pour tous les conseils qu'il m'a donnés durant ma thèse et d'avoir accepté de faire partie de mon Comité de Suivi Individuel.

Je tiens à remercier particulièrement Jérôme PASQUET pour toutes nos discussions et ses conseils qui m'ont accompagné durant mon stage de Master ainsi que durant ma thèse.

Merci également à Sébastien VILLON pour m'avoir supporté de l'IUT jusqu'à la fin de ma thèse. Apprendre à tes côtés durant toutes ses années a été un réel plaisir.

Pour poursuivre, je tiens à remercier l'ensemble de l'équipe ICAR qui m'a permis de travailler dans d'excellentes conditions. Cette équipe est composée de personnes formidables qui font en sorte que l'on puisse travailler dans la bonne humeur et dans les meilleures conditions possibles. Je n'oublierai jamais ces années passées à leurs côtés, les C&C, les repas, toutes les activités et les discussions autour de la machine à café.

Pour finir, je tiens à remercier ma famille et tous mes proches qui m'ont toujours encouragé et soutenu. Un grand merci à Nicole, merci d'avoir réussi à me supporter durant ma thèse et les moments difficiles et de m'avoir toujours soutenu.

Abstract

This thesis addresses problems related to the localization and recognition of urban objects in multi-source images (optical, infrared, terrain model) of very high precision acquired by air.

Urban objects (lamp posts, poles, car, tree...) have dimensions, shapes, textures and very variable colors. They can be glued to each other and are small with respect to the size of an image. They are present in large numbers but can be partially hidden. All this makes urban objects difficult to identify with current image processing techniques.

First, we compared traditional learning approaches, consisting of two stages - extracting features through a predefined descriptor and using a classifier - to deep learning approaches and more precisely Convolutional Neural Networks (CNN). CNNs give better results but their performances are not sufficient for industrial use. We therefore proposed two contributions to increase performance.

The first is to efficiently combine data from different sources. We compared a naive approach that considers all sources as components of a multidimensional image to an approach that merges information within CNN itself. For this, we have processed the different information in separate branches of the CNN.

For our second contribution, we focused on the problem of incomplete data. Until then, we considered that we had access to all the sources for each image but we can also place ourselves in the case where a source is not available or usable. We have proposed an architecture to take into account all the data, even when a source is missing in one or more images. We evaluated our architecture and showed that on an enrichment scenario, it allows to have a gain of more than 2% on the F-measure.

The proposed methods were tested on a public database. They aim to be integrated into a Berger-Levrault company software in order to enrich geographic databases and thus facilitate the management of the territory by local authorities.

Résumé

Cette thèse aborde des problèmes liés à la localisation et la reconnaissance d'objets urbains dans des images multi-sources (optique, infrarouge, Modèle Numérique de Surface) de très haute précision acquises par voie aérienne.

Les objets urbains (lampadaires, poteaux, voitures, arbres...) présentent des dimensions, des formes, des textures et des couleurs très variables. Ils peuvent être collés les uns aux autres et sont de petite taille par rapport à la dimension d'une image. Ils sont présents en grand nombre mais peuvent être partiellement occultés. Tout ceci rend les objets urbains difficilement identifiables par les techniques actuelles de traitement d'images.

Dans un premier temps, nous avons comparé les approches d'apprentissage classiques, composées de deux étapes - extraction de caractéristiques par le biais d'un descripteur prédéfini et utilisation d'un classifieur - aux approches d'apprentissage profond (Deep Learning), et plus précisément aux réseaux de neurones convolutionnels (CNN). Les CNN donnent de meilleurs résultats mais leurs performances ne sont pas suffisantes pour une utilisation industrielle. Nous avons donc proposé deux améliorations.

Notre première contribution consiste à combiner de manière efficace les données provenant de sources différentes. Nous avons comparé une approche naïve qui consiste à considérer toutes les sources comme des composantes d'une image multidimensionnelle à une approche qui réalise la fusion des informations au sein même du CNN. Pour cela, nous avons traité les différentes informations dans des branches séparées du CNN. Nous avons ainsi montré que lorsque la base d'apprentissage contient peu de données, combiner intelligemment les sources dans une phase de pré-traitement (nous combinons l'optique et l'infrarouge pour créer une image NDVI) avant de les donner au CNN améliore les performances.

Pour notre seconde contribution, nous nous sommes concentrés sur le problème des données incomplètes. Jusque-là, nous considérions que nous avions accès à toutes les sources pour chaque image mais nous pouvons aussi nous placer dans le cas où une source n'est pas disponible ou utilisable pour une image. Nous avons proposé une architecture permettant de prendre en compte toutes les données, même lorsqu'il manque une source sur une ou plusieurs images. Nous avons évalué notre architecture et montré que sur un scénario d'enrichissement, cette architecture permet d'obtenir un gain de plus de 2% sur la F-mesure.

Les méthodes proposées ont été testées sur une base de données publique. Elles ont pour objectif d'être intégrées dans un logiciel de la société Berger-Levrault afin d'enrichir les bases de données géographiques et ainsi faciliter la gestion du territoire par les collectivités locales.

Table des matières

Liste des figures

Liste des tableaux

Chapitre 1

Introduction

Sommaire

1.1 Présentation du contexte applicatif et positionnement de la thèse

Les travaux de cette thèse ont été réalisés dans le cadre d'une convention CIFRE entre la société Berger-Levrault et l'équipe ICAR du Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM).

La société Berger-Levrault [1] est experte du droit des administrations publiques dans les domaines de la santé, de l'éducation, du social, du sanitaire et de la gestion de territoire. Elle a vu le jour en 1676 et était à l'origine une maison d'édition d'ouvrages. Maintenant son activité principale (87% du chiffre d'affaire) est l'édition de logiciel. Dans le cadre de la gestion du territoire, Berger-Levrault met notamment à la disposition des collectivités locales, une application informatique appelée ATAL [2]. Autour d'un tronc commun, une grande diversité de modules métiers est proposée selon le domaine d'activité concerné : du référencement du patrimoine, aux bâtiments, espaces verts, à la gestion du parc automobiles et carburants, des fluides... Toute la gestion administrative et financière est intégrée : interventions, planification, management, état des stocks et des commandes, transferts en comptabilité... Certains modules métiers nécessitent une base de données voire un système d'information géographique (SIG) afin de permettre aux clients de créer des requêtes interactives, d'analyser, d'éditer ou même modifier des informations spatiales.

Le but de cette thèse est la localisation d'objets urbains en utilisant des données aériennes multi-sources telles que des images obtenues par caméra multispectrales ou des relevés LiDAR (*Light Detection And Ranging*) permettant d'enrichir les base de données de l'application ATAL. Les objets urbains (lampadaires, poteaux, voitures, arbres...) peuvent se caractériser comme des objets relativement petits, nombreux, de forme et d'apparence variable. Un exemple des données utilisées et d'objets urbains est donné en figure 1.1, avec à gauche le modèle numérique surfacique et à droite l'image multispectrale correspondante. Nous avons encadré différents objets urbains sur l'image multispectrale. D'autre part, un objet urbain peut être soumis à des occultations partielles mais aussi à la présence d'ombres, comme nous pouvons le voir dans la figure 1.2. En particulier, dans le cas des arbres, il est très courant qu'ils soient occultés par d'autres arbres ou bien partiellement ou entièrement à l'ombre lorsqu'ils sont à proximité d'objets de taille plus importante tels que des bâtiments ou bien d'autres arbres.

En décembre 2014, une enquête sur la gestion des arbres en ville [3] a montré que le coût moyen de l'entretien d'un arbre urbain par an est de 24 euros. Ainsi, en 2014, la ville de Lyon comptait 58 000 arbres, ce qui correspond à un coût d'entretien de plus de 1 390 000 euros. De plus, le fait d'arracher un arbre et d'en replanter un nouveau coûte très cher

1. https://www.berger-levrault.com
2. http://www.aductis.com
3. https://www.plante-et-cite.fr/data/fichiers_ressources/patrimoine_arbore_lyon.pdf

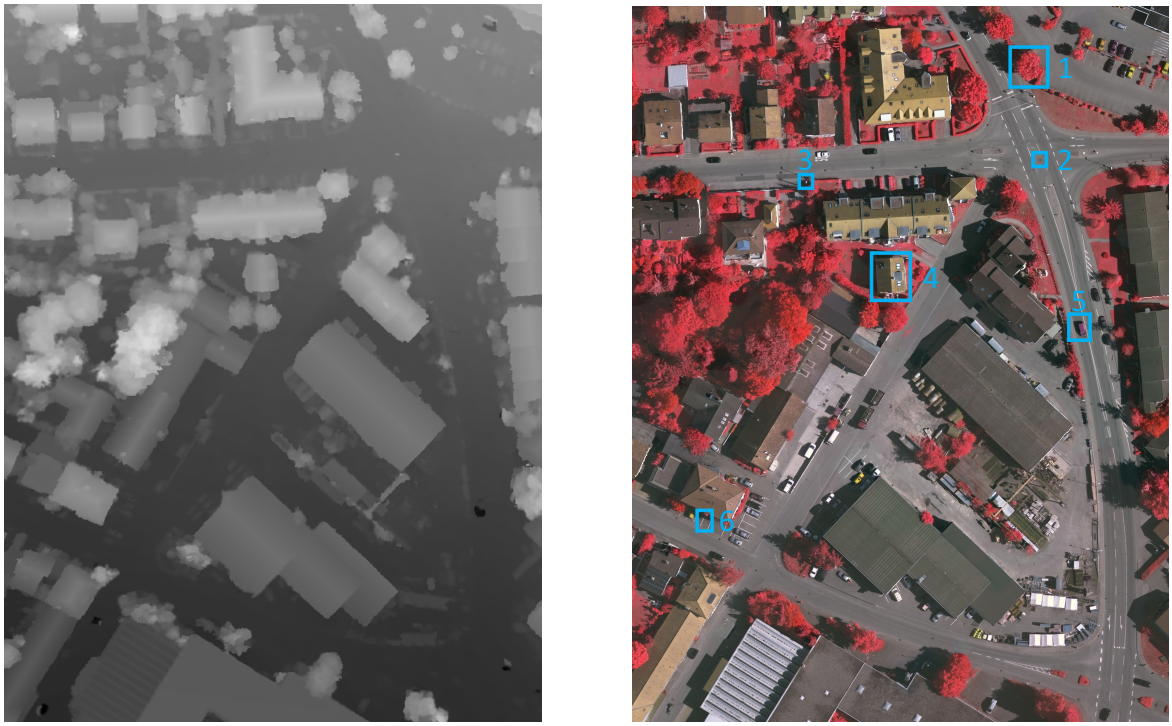


FIGURE 1.1 – Images aériennes multi-sources, avec à gauche une image en niveaux de gris du modèle numérique surfacique, et à droite l'image multispectrale correspondante. Les cadres dans l'image de droite présentent différents objets urbains : 1) un arbre 2) une bouche d'égout 3) un lampadaire 4) une maison 5) une voiture et 6) une place de parking.

FIGURE 1.2 – Image aérienne : illustration des cas d'occultation et d'ombre d'un arbre avec un bâtiment et d'un arbre avec un autre arbre.

aux communes, c'est pour cette raison qu'il est nécessaire de planifier des diagnostics réguliers afin de vérifier l'état de santé des arbres. La planification des diagnostics et des interventions sur les arbres nécessite de connaître leur localisation, c'est pour cette raison que nous nous sommes concentrés sur les arbres en milieu urbain.

Dans cette thèse nous nous sommes donc focalisés sur la localisation de l'objet urbain arbre. Des travaux préliminaires en 2011 et 2012 ont été réalisés sur la localisation de tombes à partir d'images aériennes dans le cadre d'une collaboration entre l'équipe ICAR

du LIRMM et la société Berger-Levrault. Ils ont permis de tester des méthodes connues de la littérature à savoir l'approche de Viola et Jones [Viola et Jones, 2001] et l'approche de Aldavert et al. [Aldavert *et al.*, 2010] et de montrer les difficultés de cette tâche. À la suite de ce constat, la thèse CIFRE de Jérôme Pasquet dont le sujet était "Modélisation, détection et classification d'objets urbains à partir d'images photographiques aériennes" a débutée en 2013. Au cours de cette thèse, Jérôme Pasquet a proposé d'utiliser un réseau de Séparateurs à Vaste Marge (SVM) permettant d'améliorer les performances sur la détection d'objets urbains en comparaison aux approches classiques de détection [Pasquet *et al.*, 2015]. Ce réseau a permis d'améliorer les performances de détection mais avec un coût calculatoire relativement important. Afin de réduire le coût calculatoire de cette nouvelle approche, il a proposé d'utiliser un chemin d'activation qui a permis de réduire la complexité calculatoire sans perdre en efficacité [Pasquet *et al.*, 2015]. Sa dernière contribution a été de combiner son réseau de SVM avec un réseau de neurones convolutifs [Pasquet *et al.*, 2016]. La combinaison de ces deux approches, en plus du chemin d'activation, lui a permis d'avoir un gain d'environ 8% sur les performances tout en réduisant le coût d'activation de son réseau d'au plus 97%.

Les travaux présentés ici se placent dans la continuité de la thèse de Jérôme Pasquet, c'est-à-dire sur le problème de localisation d'objets urbains mais en utilisant cette fois des données provenant de sources multiples (images multispectrales et relevés LiDAR). L'idée étant d'enrichir notre base d'images optiques avec des données additionnelles qui peuvent être sous forme d'images ou bien de nuages de points. Dans le cadre de cette thèse, nous nous sommes concentrés sur le cas d'informations additionnelles sous forme d'image, notamment en ajoutant des connaissances LiDAR par le biais d'une carte d'élévations issue du Modèle Numérique Surfacique (MNS) ainsi que du proche infrarouge. Le fait de combiner des données provenant de sources différentes engendre de nouveaux problèmes. En effet les problèmes de fusion de données et d'évaluation ne sont pas triviaux. Il y a aussi le problème de données incomplètes car dans une application réelle, il est courant de ne pas avoir accès à toutes les sources. Dans la continuité de la thèse de Jérôme Pasquet, nous avons choisi d'utiliser dans nos travaux des réseaux de neurones convolutifs.

Notre thèse se situe donc à l'intersection entre la télédétection et le traitement des données visuelles. La télédétection ou *remote sensing* en anglais, désigne un ensemble de techniques qui permettent d'étudier à distance des objets ou des phénomènes.

1.2 La télédétection

Pour définir la télédétection de manière plus opérationnelle et plus précise, on peut considérer la télédétection comme un ensemble de techniques qui permettent la mesure ou l'acquisition d'informations d'un phénomène ou d'un objet par des mesures effectuées à distance, sans contact matériel avec celui-ci. La télédétection englobe l'acquisition et l'enregistrement de l'information ainsi que l'analyse et le traitement de l'informa-

tion enregistrée.

Ce type d'étude est réalisée à partir de drones, d'avions, de ballons ou de satellites en utilisant les propriétés du rayonnement électromagnétique émis, réfléchi ou diffusé par ce que l'on cherche à étudier (surfaces, objets, végétation...). Au début de la télédétection, le principal capteur utilisé était l'appareil photographique. Les capteurs ont depuis largement évolué, notamment avec des capteurs comme le radiomètre. Ce type de capteur permet d'enregistrer les rayonnements naturels, la lumière visible mais aussi proche infrarouge sous forme numérique. L'évolution de ces capteurs a permis de grandement améliorer la résolution spatiale, c'est-à-dire la finesse des détails des données que l'on enregistre. Par exemple, avec les satellites SPOT-6 et SPOT-7 lancés respectivement en 2012 et 2014, il est possible d'obtenir des images multispectrales avec une résolution spatiale de 6 mètres.

Dans notre cas, l'acquisition a été faite à très basse altitude (900 mètres du sol) et nous travaillons sur une résolution spatiale beaucoup plus petite, puisqu'elle est de 9 cm. De plus, nous travaillons sur de la recherche d'objets qui ne sont pas étudiés en télédétection classique. En effet, généralement la télédétection se concentre sur des tâches comme la la classification de scènes [Liu *et al.*, 2018], le détourage d'objets [Sun *et al.*, 2010] ou bien la segmentation sémantique [Yang *et al.*, 2017].

1.3 Les outils actuels pour l'analyse des données visuelles

De nos jours, les données visuelles sont devenues omniprésentes dans notre quotidien. En effet, nous avons pu assister ces dernières années à une explosion des informations telles que les images et les vidéos par le biais des réseaux sociaux. Par exemple, en février 2017 Google a annoncé que plus d'un milliard d'heures de vidéo sont visionnées sur Youtube chaque jour[4]. Facebook compte 240 milliards d'images et 350 millions d'images sont ajoutées par jour[5]. Avec cette augmentation brutale des données visuelles, les problèmes liés à l'analyse des données visuelles et notamment l'apprentissage automatique, sont devenus des problèmes cruciaux. En effet, il est devenu important de pouvoir automatiser leur analyse afin de déterminer le contenu de ces données. C'est pour cette raison que de nombreuses entreprises telles que Facebook et Google ont lancé de nombreux projets de recherche et de développement afin d'aboutir sur des produits et des services dédiés entre autres à l'analyse d'image.

On peut distinguer trois principaux objectifs :

— la classification,

— la détection / localisation,

— la segmentation.

4. https://www.blogdumoderateur.com/chiffres-youtube/
5. https://www.blogdumoderateur.com/chiffres-facebook/

L'objectif de la classification est de donner la ou les catégories sémantiques du contenu des images. Lorsque l'on souhaite réaliser une tâche de détection, l'objectif est de donner la ou les catégories sémantiques des objets contenus dans les images, mais aussi d'être capable de donner leurs positions dans les images. Enfin, la segmentation consiste à déterminer la catégorie sémantique de chaque pixel lorsqu'il s'agit d'une segmentation sémantique, c'est-à-dire lorsque nous connaissons le nombre et les catégories sémantiques possibles (sinon le but est de partitionner les images selon un critère d'homogénéité). Les problèmes de classification, de détection et de segmentation sont des problèmes complexes. De plus, pour une automatisation acceptable, il faut que les méthodes utilisées soient très efficaces, c'est-à-dire très proches de 100% de réussite.

Le problème de classification est très facile pour un humain, cependant pour une machine ce problème n'est pas si simple car elle ne "voit" que des nombres (c'est-à-dire les valeurs des pixels) sans aucune sémantique. Le but est donc de faire correspondre ces nombres avec une ou plusieurs catégories. Pour faire cela, il faut être capable de comprendre la sémantique d'une image en se basant uniquement sur son contenu visuel. En se basant uniquement sur une représentation bas niveau de l'image, c'est-à-dire les pixels, il est impossible de déduire une sémantique. Pour pouvoir classifier des images automatiquement, nous devons extraire des représentations de haut niveau à partir des images. De plus ces représentations de haut niveau doivent aussi être invariantes aux variations intra-classe tout en restant sensibles aux variations inter-classe.

Jusqu'aux années 2000, la majorité des méthodes d'extraction de représentation de haut niveau étaient basées sur des caractéristiques qui étaient construites "à la main". Cette famille de méthodes nécessite un expert du domaine pour concevoir un extracteur de caractéristiques, aussi appelé descripteur, qui transforme les pixels de l'image brute en un vecteur de caractéristiques. Par exemple, si le but est de différencier des animaux, il peut être intéressant d'utiliser des caractéristiques comme les formes, les couleurs et la texture.

Depuis la victoire de Krizhevsky et al. en 2012 [Krizhevský *et al.*, 2012] à la compétition ImageNet Large Scale Visual Recognition Challenge (ILSVRC), les réseaux convolutionnels profonds (CNN), c'est-à-dire les réseaux de neurones composés de plusieurs couches de convolution, sont devenus l'état de l'art en reconnaissance visuelle. Les CNNs sont depuis 2012 largement utilisés, bien que la famille des réseaux de neurones existe depuis le milieu du XXe siècle.

Le succès des algorithmes d'apprentissage automatique (Machine Learning) basés sur des réseaux profonds a commencé en 2006 grâce à Hinton et al. [Hinton *et al.*, 2006] qui ont introduit une nouvelle méthode de pré-entraînement pour les Deep Belief Networks (DBNs). Cependant des réseaux de neurones utilisant des convolutions [LeCun *et al.*, 1990; Lecun *et al.*, 1998] avaient déjà fait leurs preuves sur le problème de classification d'images de chiffres manuscrits sur la base de données MNIST [6]. Ils ont obtenu de meilleurs résultats qu'un classifieur qui était alors considéré comme un des meilleurs, le

6. http ://yann.lecun.com/exdb/mnist/

Séparateur à Vaste Marge [Boser *et al.*, 1992; Cortes et Vapnik, 1995; Vapnik, 1995] (SVM). L'accélération des calculs grâce à l'utilisation des GPUs et l'apparition de certaines méthodes, comme la méthode du *dropout* [Srivastava *et al.*, 2014] ont permis de réduire le temps d'apprentissage des réseaux et d'augmenter leurs performances. Les architectures de ces nouveaux réseaux de neurones sont profondes, c'est-à-dire avec plusieurs couches, et permettent de modéliser les données avec un haut niveau d'abstraction [Bengio *et al.*, 2013] tout en réalisant la classification.

En conclusion, l'objectif de la thèse est la détection et la localisation d'objets urbains dans des images aériennes multi-sources. Dans notre cas, nous nous sommes intéressés à la détection d'arbres en milieu urbain. Les arbres peuvent avoir des tailles ou bien des formes variées. Ils peuvent être très proches les uns des autres mais aussi être occultés par d'autres arbres ou bien d'autres objets urbains tels que des bâtiments.

Afin de localiser les arbres, nous avons fait le choix d'utiliser une approche objet, cela signifie que nous cherchons à détecter des zones qui contiennent un objet urbain de type arbre. Pour déterminer les zones contenant un arbre, nous avons choisi de réaliser nos travaux avec des réseaux de neurones convolutionnels.

1.4 Présentation de la base de données

Nous présentons dans cette section la base de données publique qui a été utilisée tout au long de cette thèse et qui nous a permis d'avoir un suivi des résultats. La base de données Vaihingen est fournie par la Société allemande de photogrammétrie, de télédétection et de géoinformation (DGPF) [Cramer, 2010] [7]. Cette base de données a été acquise dans la ville de Vaihingen en Allemagne. La figure 1.4 montre la zone d'acquisition et l'ensemble des images qui composent cette base de données. Elle contient trois types d'images :

— 33 images d'environ 2000 × 2500 pixels avec les canaux proche infrarouge, rouge et vert (PirRV). Les données PirRV ont été acquises à l'aide d'une caméra numérique de cartographie DMC Intergraph / ZI volant à 900m de hauteur, avec une résolution au sol de 9 cm, par la société RWE Power le 24 juillet et le 6 août 2008.

— 33 images en niveaux de gris correspondant au modèle numérique surfacique (MNS). Le MNS avec résolution au sol de 9 cm a été extrait à partir de méthodes d'appariement d'images.

— 16 images avec les canaux rouge, vert et bleu (RVB) correspondant à la vérité terrain annotée **par pixel** avec :

  — en bleu la classe *building*,

  — en jaune la classe *car*,

  — en rouge la classe *clutter / background*,

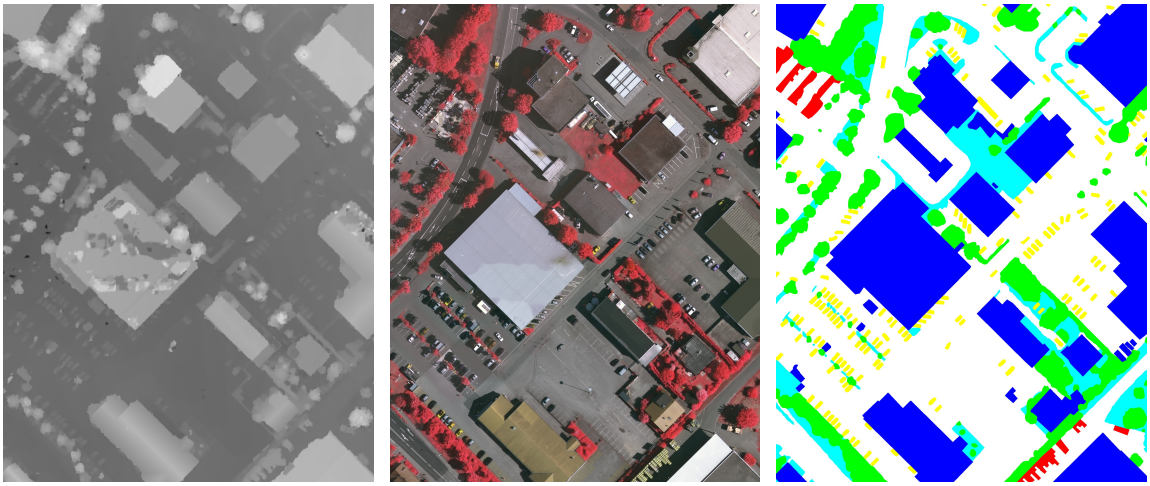
7. http ://www.ifp.uni-stuttgart.de/dgpf/DKEP-Allg.html.

— en turquoise la classe *low vegetation*,

— en vert la classe *tree*,

— en blanc la classe *impervious surfaces*.

La figure 1.3 présente une zone de la base de données avec le MNS, une image PirRV et la vérité terrain par pixel.



(a) (b) (c)

FIGURE 1.3 – Les différents types d'images présents dans la base de données Vaihingen : (a) Modèle Numérique Surfacique (MNS), (b) Image avec les canaux Proche Infrarouge, Rouge et Vert, (c) Vérité terrain par pixel.

Cette base de données comprend trois secteurs (voir figure 1.4). Le premier secteur est situé dans le centre de la ville de Vaihingen, il se caractérise par des bâtiments historiques ayant des formes assez complexes, mais a également quelques arbres. Le deuxième secteur est caractérisé par quelques bâtiments résidentiels élevés qui sont entourés d'arbres. Le troisième secteur est résidentiel avec de petites maisons individuelles. Le tableau 1.1 récapitule le contenu de la base de données, avec la taille de chaque image et si oui ou non la zone a une vérité terrain correspondante.

Les images de cette base de données ont été utilisées pour deux tâches différentes :

— La segmentation sémantique. Pour cette tâche que nous avons utilisée pour notre deuxième contribution présentée chapitre 5, nous avons utilisé la vérité terrain fournie avec la base de données, c'est-à-dire l'annotation par pixel.

— La localisation et la détection d'arbres. Pour cette tâche nous avons utilisé l'ensemble des données Pir, RV et MNS. Il nous a fallu dans un premier temps construire la vérité terrain **objet** puisque nous ne disposions que de la vérité terrain pixel. Ainsi, nous avons dû annoter chaque arbre présent dans les images pour qu'ils possèdent une boîte englobante, c'est-à-dire les coordonnées du rectangle englobant l'arbre. Après annotation, nous avons 1 500 arbres localisés manuellement. Un exemple d'annotation est présenté figure 1.5. Nous avons par la suite ajouté

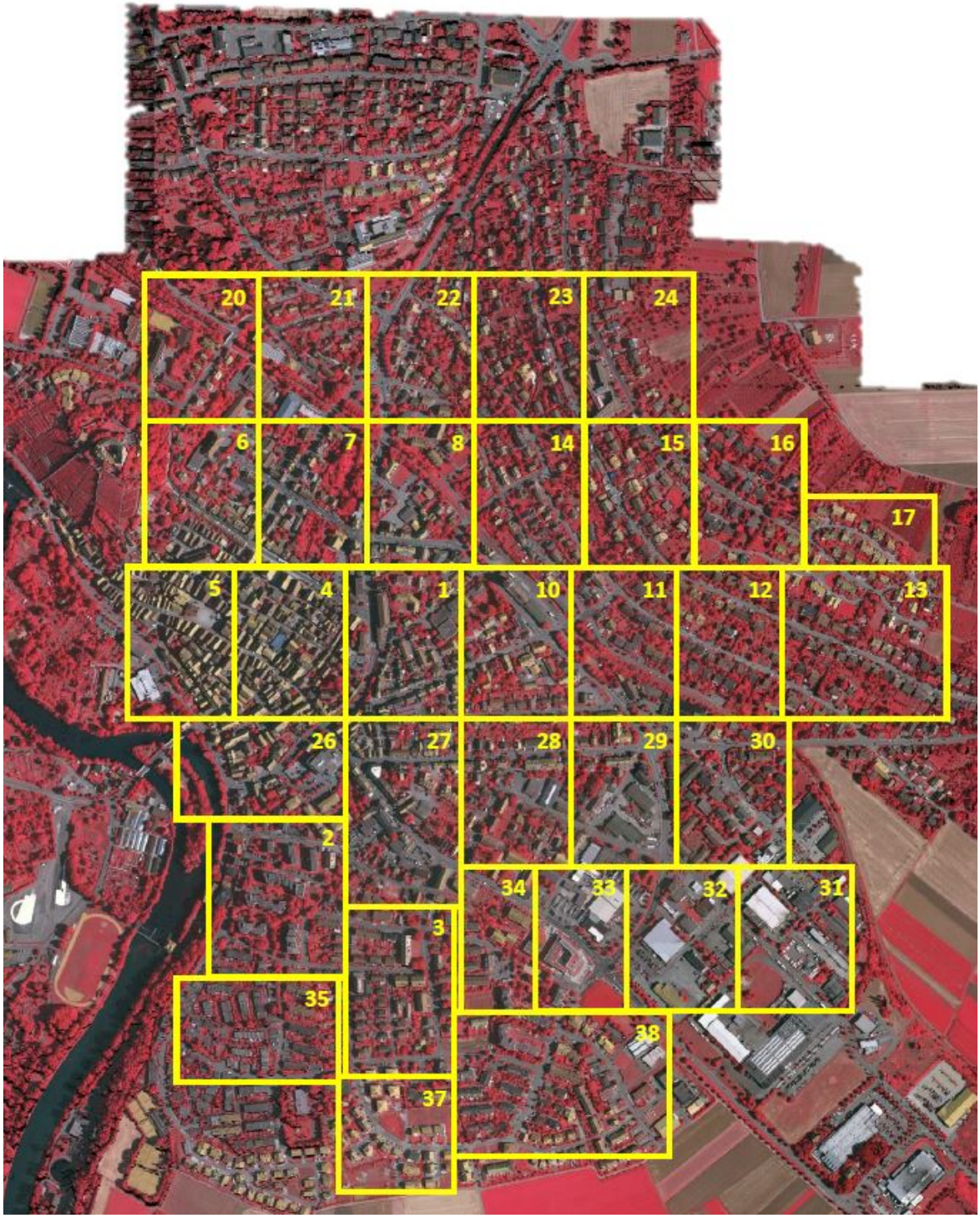


FIGURE 1.4 – Image montrant l'ensemble de l'acquisition de la base Vaihingen ainsi que les contours de chaque zone. Le numéro des zones est indiqué en haut à droite dans les rectangles jaune.

deux autres classes, qui sont la classe *building* et la classe *low vegetation*. Pour extraire des imagettes [8] de la classe *building* et de la classe *low vegetation*, nous nous sommes servis de la vérité terrain pixel. Pour extraire des imagettes de la classe

8. Une imagette est une partie d'une image représentant un objet correspondant à une classe (un arbre, un bâtiment...). Dans notre cas, les imagettes sont redimensionnées à une dimension de 64×64 pixels.

PirRV / MNS	Dimensions	Vérité Terrain
1	1919×2569	x
2	2428×2767	
3	2006×3007	x
4	1887×2557	
5	1887×2557	x
6	1887×2557	
7	1887×2557	x
8	1887×2557	
10	1887×2557	
11	1893×2566	x
12	1922×2575	
13	2818×2558	x
14	1919×2565	
15	1919×2565	x
16	1919×2565	
17	2336×1281	x

PirRV / MNS	Dimensions	Vérité Terrain
20	1866×2315	
21	1903×2546	x
22	1903×2546	
23	1903×2546	x
24	1903×2546	
26	2995×1783	x
27	1917×3313	
28	1917×2567	x
29	1917×2563	
30	1934×2563	x
31	1980×2555	
32	1980×2555	x
33	1581×2555	
34	1388×2555	x
35	2805×1884	
37	1996×1995	x
38	3816×2550	

TABLEAU 1.1 – Tableau récapitulatif de la base de données Vaihingen avec les dimensions de chaque image. Lorsque nous avons la vérité terrain d'une image, nous avons mis une croix dans la colonne "vérité terrain".

FIGURE 1.5 – Exemples d'annotations réalisés sur la base de données Vaihingen avec en bleu les boîtes englobantes déterminées par l'annotateur.

building, nous avons extrait des imagettes dont la zone est entièrement bleue dans l'image correspondant à la vérité terrain. Nous avons appliqué le même processus pour extraire les imagettes de la classe *low vegetation*. Nous avons extrait au total 14 666 imagettes *building*, 25 787 imagettes *low vegetation* et 41 075 imagettes *back-*

ground. La classe *background* correspond à tout ce qui n'est pas dans les classes *arbre*, *low vegetation* et *building*.

1.5 Plan de la thèse

La suite du manuscrit est composée d'une première partie sur l'état de l'art et ensuite d'une deuxième partie consacrée à nos contributions.

Dans le chapitre 2, nous définirons l'approche par apprentissage automatique. L'apprentissage automatique classique est composé de deux étapes, avec d'abord l'utilisation d'un descripteur, et ensuite l'utilisation d'un classifieur. Dans un premier temps, nous aborderons les descripteurs, notamment en présentant les descripteurs par histogrammes et les descripteurs par gradient. Dans un deuxième temps nous présenterons les classifieurs SVM et par forêts aléatoires. Ensuite nous détaillerons le fonctionnement et la construction d'un réseau de neurones convolutionnels. Puis nous verrons leur utilisation pour la détection et la segmentation sémantique. Nous terminerons par aborder la gestion des données multi-sources et des données incomplètes et/ou manquantes.

Dans le chapitre 3, nous présenterons une étude préliminaire visant à comparer les méthodes d'apprentissage automatique classiques en deux étapes à une approche par réseau de neurones convolutionnels. Dans ce chapitre, nous expliquerons la méthodologie et les mesures utilisées qui nous ont permis de comparer ces deux approches.

Le chapitre 4 présentera une solution que nous proposons afin de traiter des données multi-sources par le biais d'un réseau convolutif. Enfin, dans le chapitre 5 nous proposons une nouvelle architecture de réseau permettant la gestion des données manquantes et incomplètes.

Finalement, nous conclurons le manuscrit par le chapitre 6, dans lequel nous résumerons nos contributions et nous proposerons plusieurs perspectives à court et long terme.

Chapitre 2

État de l'art

Sommaire

2.1 Apprentissage automatique en deux étapes

Dans cette section, nous décrivons plus spécifiquement les approches d'apprentissage automatique en deux étapes. Nous n'aborderons pas ici les algorithmes explicites, c'est-à-dire les algorithmes ne nécessitant pas d'apprentissage comme dans [Niigaki *et al.*, 2012; Pasquet *et al.*, 2016] où les auteurs utilisent une recherche de motifs circulaires (cas des arbres ou des plaques d'égoût) dans des images. Dans le cas d'un apprentissage automatique classique ou par *deep learning*, nous devons utiliser une base de données d'entraînement afin de fixer les valeurs des paramètres de ces méthodes. La figure 2.1 illustre les différentes étape de l'apprentissage automatique classique et la grande différence qu'il y a avec le *deep learning*. En effet, même si ces deux approches font partie de la même famille, celle de l'apprentissage automatique, dans le cas du *deep learning*, l'extraction de caractéristiques et la prédiction (la partie rouge de la figure 2.1) sont faites en même temps alors que ces étapes sont bien distinctes dans le cas d'un apprentissage automatique en deux étapes.

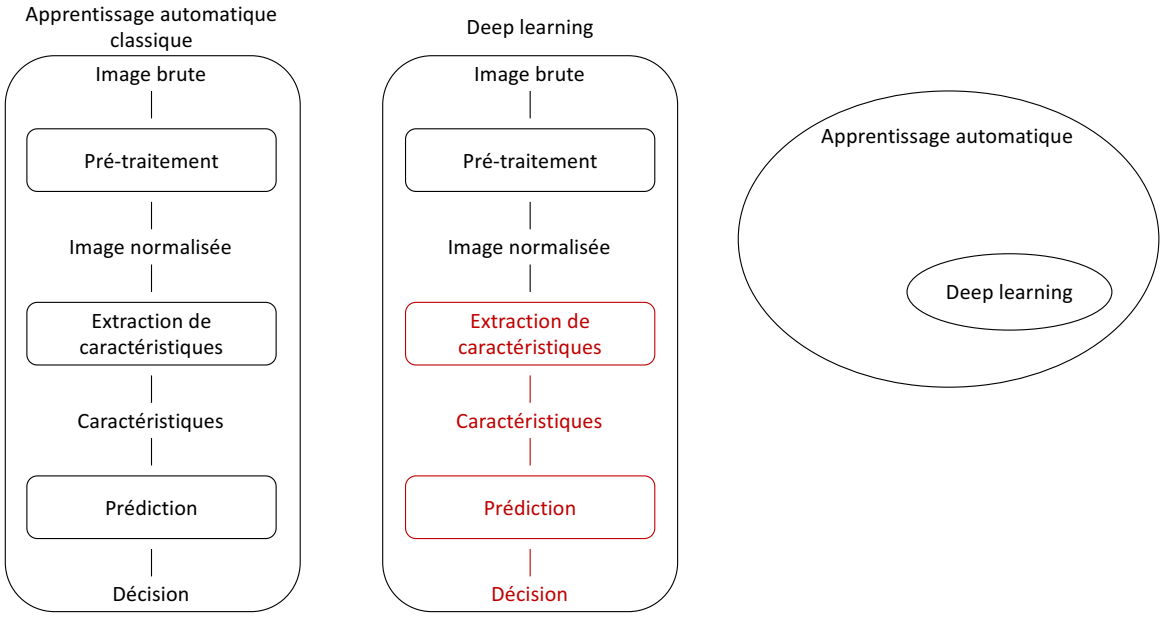


FIGURE 2.1 – Présentation des différences entre l'apprentissage classique et le *deep learning*.

Cette section est décomposée en deux parties. Ces deux parties correspondent aux deux étapes principales permettant de réaliser un apprentissage automatique classique, c'est-à-dire l'extraction de caractéristiques en utilisant un descripteur et la prédiction avec l'utilisation d'un classifieur. Dans la section 2.1.1, nous présentons des descripteurs visuels, et dans la section 2.1.2 nous présentons des classifieurs.

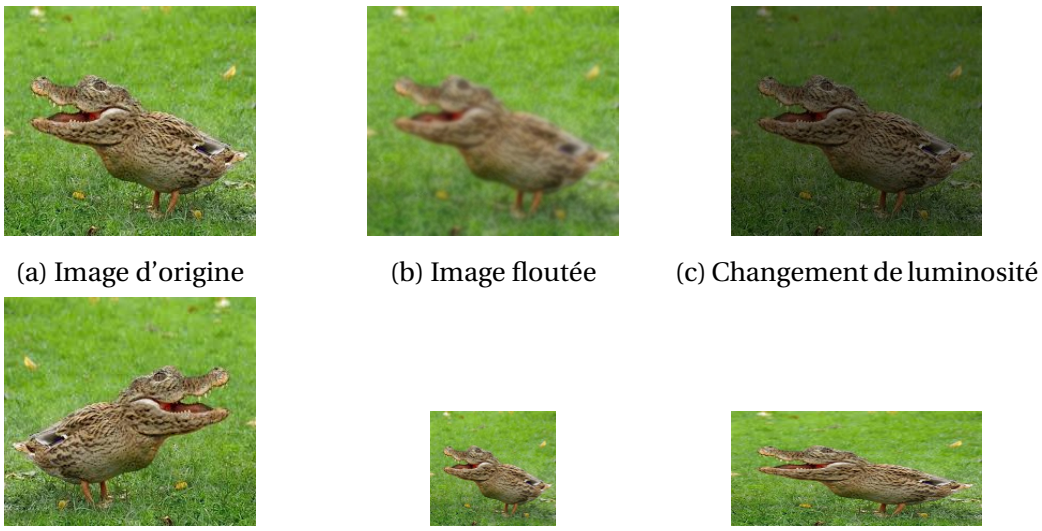
2.1.1 Les descripteurs

Un descripteur est un vecteur de valeurs qui décrit le contenu d'une image ou bien d'une partie F d'une image. Nous ne parlerons ici que de descripteurs conçus par des

spécialistes. On appelle ces descripteurs des descripteurs *handcrafted*. On oppose souvent ces descripteurs aux descripteurs appris.

La première stratégie consiste à décrire une image dans sa totalité. Ces types de descripteurs sont très utilisés notamment pour retrouver ou bien indexer des images entières dans des bases de données [Coelho et Ribeiro, 2010; Kertész et Oy, 2011]. La deuxième stratégie consiste à décrire une image localement, cela veut dire que l'on va extraire plusieurs descripteurs dans des sous-parties d'une image. Les descripteurs locaux sont utilisés en particulier pour réaliser de la détection et de la localisation d'objets dans les images. Pour extraire ces descripteurs, on utilise une fenêtre F souvent de taille fixe, qui va parcourir l'image. Pour chaque position de la fenêtre F, on va extraire un descripteur.

Le principe général du descripteur est de représenter le contenu de la fenêtre F tout en étant le plus "robuste" possible. Par robuste, on veut dire que l'on souhaite être capable de détecter un objet même si il y a eu une déformation photométrique ou géométrique comme un changement de luminosité ou bien d'échelle. Par exemple, on voudrait que pour toutes les images de la figure 2.2 le vecteur de caractéristiques soit le même.



(a) Image d'origine (b) Image floutée (c) Changement de luminosité

(d) Miroir horizontal (e) Image redimensionnée (f) Image étirée

FIGURE 2.2 – Exemples de différentes déformations photométriques et géométriques.

Dans la suite, nous allons présenter deux types de descripteurs : les descripteurs photométriques, les descripteurs par gradients.

Les descripteurs photométriques permettent de prendre en compte l'information d'intensité lumineuse (acquisition par un capteur photographique) du contenu présent dans la fenêtre F. Nous allons présenter ici deux types de descripteurs photométriques. Le premier va utiliser l'histogramme des intensités des pixels sur chacun des canaux de l'image, et le deuxième se définit par un moment qui modélise la distribution des intensités des pixels de la fenêtre F. L'extension de ces descripteurs à la couleur peut rendre le descripteur très dépendant de l'espace couleur utilisé.

2.1.1.1 Descripteurs par histogramme

Un descripteur par histogramme consiste à construire un histogramme des intensités pour chaque canal couleur de la fenêtre F. On va donc utiliser pour chaque couleur un histogramme qui va comptabiliser les intensités des pixels pour une couleur donnée. Ensuite, l'ensemble des histogrammes vont être concaténés afin de former un seul descripteur [Gevers *et al.*, 2006]. Dans le cas de ce type de descripteur, on ne tient pas compte des relations de voisinages entre les pixels.

La perte de l'information spatiale est bien sûr un défaut majeur lorsque l'on souhaite faire de la localisation d'objets. Pour pallier ce problème, il existe une solution qui consiste à diviser F en sous-régions. Un histogramme est alors calculé et normalisé pour chacune des sous-régions de F. L'ensemble de ces histogrammes sont concaténés, par exemple en mettant les histogrammes les uns à la suite des autres dans un vecteur pour obtenir le descripteur final. Cette approche permet de garder une information spatiale puisqu'il y a plusieurs histogrammes et que ceux-ci sont concaténés.

Les descripteurs basés sur l'histogramme des intensités des pixels peuvent être très efficaces dans certains cas. Par exemple dans [Ramli *et al.*, 2008] et [Sergyan, 2008], les auteurs montrent dans leur travaux qu'utiliser des descripteurs qui se basent sur des histogrammes des intensités des pixels est très efficace pour réaliser une tâche de classification. Cependant il y a des cas où utiliser les intensités des pixels ou la couleur ne suffit pas. La figure 2.3 illustre un cas pour lequel un descripteur par histogramme n'est pas suffisant. Les images (a) et (b) de la figure sont des images qui contiennent un arbre et les images (c) et (d) ne contiennent que de l'herbe. Comme on peut le constater sur les histogrammes (e), (f), (g) et (h), il est impossible de déterminer quel histogramme représente un arbre ou bien de l'herbe.

2.1.1.2 Descripteurs par gradients

Lorsque l'on souhaite réaliser une tâche de détection et de localisation, l'information relative à la forme et aux contours des objets d'une image est très importante. Une approche simpliste de détection des contours consiste à étudier les discontinuités d'intensité des pixels par rapport à leur voisinage, le gradient peut justement nous donner cette information [Huang *et al.*, 2003; Aggarwal *et al.*, 2015]. Cette détection est généralement appliquée sur une image pré-traitée et non pas sur l'image brute. Les descripteurs par gradients permettent d'avoir une représentation de l'arrangement spatial et des discontinuités contenues dans l'image. Ces descripteurs peuvent par exemple reposer sur le calculs des dérivées en chaque pixel. Pour ce faire on doit calculer les dérivées verticales et les dérivées horizontales. On note $D^{(x)}$ les dérivées horizontales de l'image I et $D^{(y)}$ ses dérivées verticales :

$$D^{(x)} = G * I \text{ et } D^{(y)} = G^{T} * I, \tag{2.1}$$

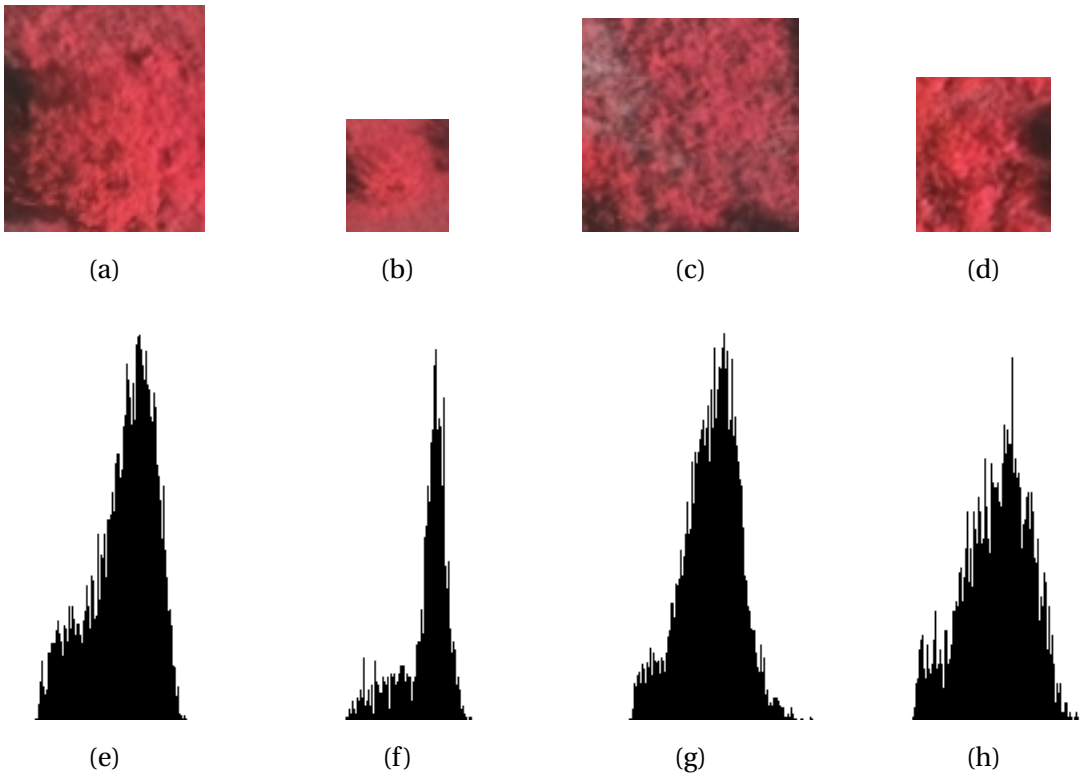


FIGURE 2.3 – Exemples d'images pour lesquelles les descripteurs par histogramme des intensités des pixels ne fonctionnent pas. Les images (a) et (b) sont des images qui contiennent des arbres, et les images (c) et (d) sont des images qui contiennent uniquement de l'herbe. Les histogrammes (e), (f), (g) et (h) correspondent respectivement aux images (a), (b), (c) et (d) transformées en niveaux de gris.

avec G le filtre dérivateur utilisé et $*$ l'opérateur de convolution. On peut par exemple utiliser le filtre de Sobel [Sobel, 1990]. C'est ce filtre qui a été utilisé dans [Huang *et al.*, 2003; Aggarwal *et al.*, 2015]. Le filtre de Sobel se définit pour la direction horizontale comme suit :

$$S_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}. \tag{2.2}$$

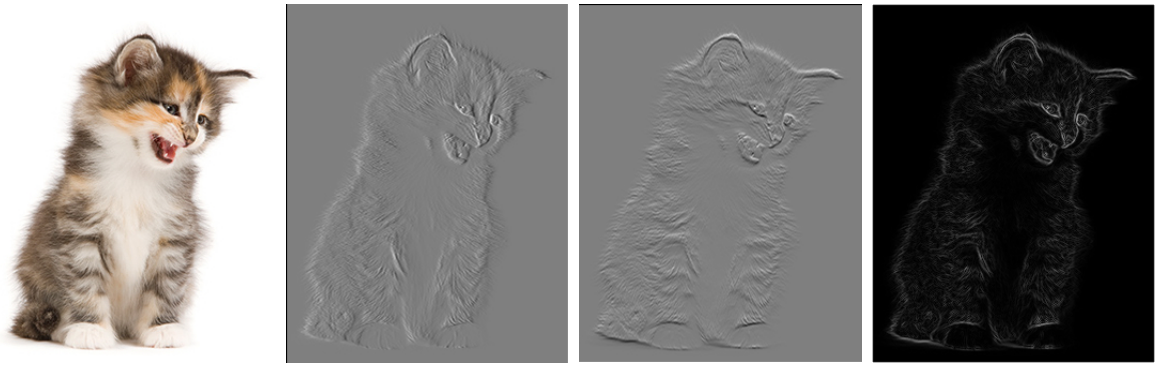
C'est à partir des images dérivées que l'on va calculer l'orientation du gradient et le module du gradient en chaque pixel. Le module G et l'orientation θ sont définis comme suit en un pixel à la position (x, y) :

$$G_{x,y} = \sqrt{(D^{(x)}_{x,y})^2 + (D^{(y)}_{x,y})^2}, \tag{2.3}$$

$$\theta_{x,y} = \arctan\left(\frac{D^{(y)}_{x,y}}{D^{(x)}_{x,y}}\right). \tag{2.4}$$

La figure 2.4 présente un exemple qui illustre les images des dérivées horizontales et verticales ainsi que l'image du module du gradient. Dans cet exemple, l'image (a) correspond à l'image d'origine, l'image (b) correspond à l'image des dérivées horizontales, l'image

(c) représente l'image des dérivées verticales et enfin l'image (d) correspond à l'image du module du gradient.



(a) (b) (c) (d)

FIGURE 2.4 – Illustration des dérivées avec (a) l'image d'origine, (b) l'image des dérivées horizontale $D^{(x)}$, (c) l'image des dérivées verticales $D^{(y)}$ et (d) l'image du module des gradients.

La figure 2.5 présente le cas où l'on utilise le descripteur par gradients sur des images de la base de données présentée chapitre 1. On peut voir très facilement les arbres dans l'image (a), cependant il est difficile de les distinguer dans les images des dérivées (b) et (c) et dans l'image du module des dérivées (d). Même si on arrive à deviner la forme d'un arbre sur les images dérivées et sur l'image du module des gradients, il est impossible de voir qu'il y a deux arbres dans cette image.

La deuxième ligne de la figure 2.5 présente le cas où un arbre est difficilement visible pour l'œil humain. Dans ce cas là, on peut voir dans les images des dérivées (f) et (g), et dans l'image du module des dérivées (h) qu'il est impossible de voir un arbre.

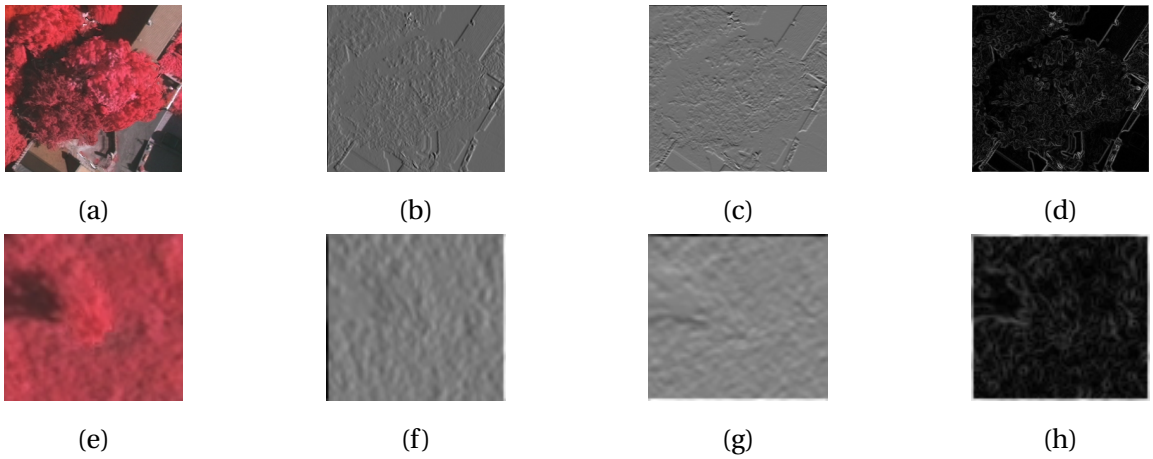
Ces exemples nous montrent les limites du descripteur par gradients. Ce type de descripteur peut être efficace dans des cas où les objets que l'on souhaite étudier ont des formes bien différentiables et distinctes par rapport à ce qui l'entoure. Lorsque que comme dans la figure 2.5, l'objet en question se confond avec ce qui l'entoure ou bien lorsqu'il y a des occultations, ce descripteur perd en efficacité.

2.1.1.3 Le descripteur SIFT

Le descripteur SIFT (*Scale-Invariant Feature Transform*), est un descripteur qui est très utilisé pour son invariance et sa capacité à décrire des objets comme des personnes. Ce descripteur a été développé en 1999 par le chercheur David Lowe [Lowe, 1999] et amélioré en 2004 par Lowe [Lowe, 2004].

L'idée de ce descripteur est de caractériser le contenu visuel d'une image de la façon la plus indépendante possible de l'échelle, du cadrage, de l'angle d'observation et de la luminosité lorsque les images sont en niveaux de gris. Pour ce faire, l'algorithme pour obtenir un descripteur SIFT est le suivant :

Les étapes 1, 2 et 3 de l'algorithme permettent de sélectionner les points clés. Détaillons maintenant les différentes étapes de la construction de ce descripteur :



(a) (b) (c) (d)

(e) (f) (g) (h)

FIGURE 2.5 – Illustration des dérivées sur des patchs de la base de données décrite dans le chapitre 1. Sur cet exemple nous montrons les limites des descripteurs par gradients. L'image (a) contient deux arbres facilement détectables par l'œil humain, cependant on peut constater qu'il n'est pas aisé de repérer les arbres dans les images des dérivées (b), (c) et dans l'image du module des gradients (d). Il en est de même pour l'arbre présent dans l'image (e).

Algorithme 1 : Construction du descripteur SIFT

Données : I une image en niveaux de gris

Résultat : Descripteur SIFT

1 Détection d'extrema ;
2 Localisation des points clés ;
3 Affectation d'une orientation aux points clés ;
4 Calcul du descripteur pour chaque point clé ;
5 Concaténation et normalisation des descripteurs de tous les points clés ;
6 Retourner le descripteur final ;

1. Détection d'extrema dans l'espace des échelles : dans un premier temps on génère un ensemble d'images par un filtrage à différentes échelles via la convolution du canal de l'image I par un filtre gaussien G de paramètre σ. Ce résultat est noté L et a un facteur d'échelle σ :

$$\mathrm{L}(x, y, \sigma) = \mathrm{G}(x, y, \sigma) * \mathrm{I}(x, y), \tag{2.5}$$

avec $*$ l'opération de convolution et

$$\mathrm{G}(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}. \tag{2.6}$$

Cette convolution a pour but de lisser l'image I afin que les détails qui ont un rayon inférieurs à σ soient estompés. Dans un second temps nous utilisons toutes les images lissées obtenues pour générer un autre ensemble d'images. Pour obtenir cet ensemble d'images, nous utilisons la différence de gaussiennes (DOG en anglais) qui

se définit comme suit :

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma), \tag{2.7}$$

avec k un facteur multiplicatif qui doit être déterminé par l'utilisateur, généralement $k = 2$. Ensuite, l'image gaussienne est sous-échantillonnée par un facteur de 2, et le processus est répété. Cette première étape est représentée figure 2.6. La partie gauche du schéma représente les images résultantes de la convolution de l'image d'origine avec les noyaux gaussiens G. La partie droite du schéma représente les résultats des DOG.

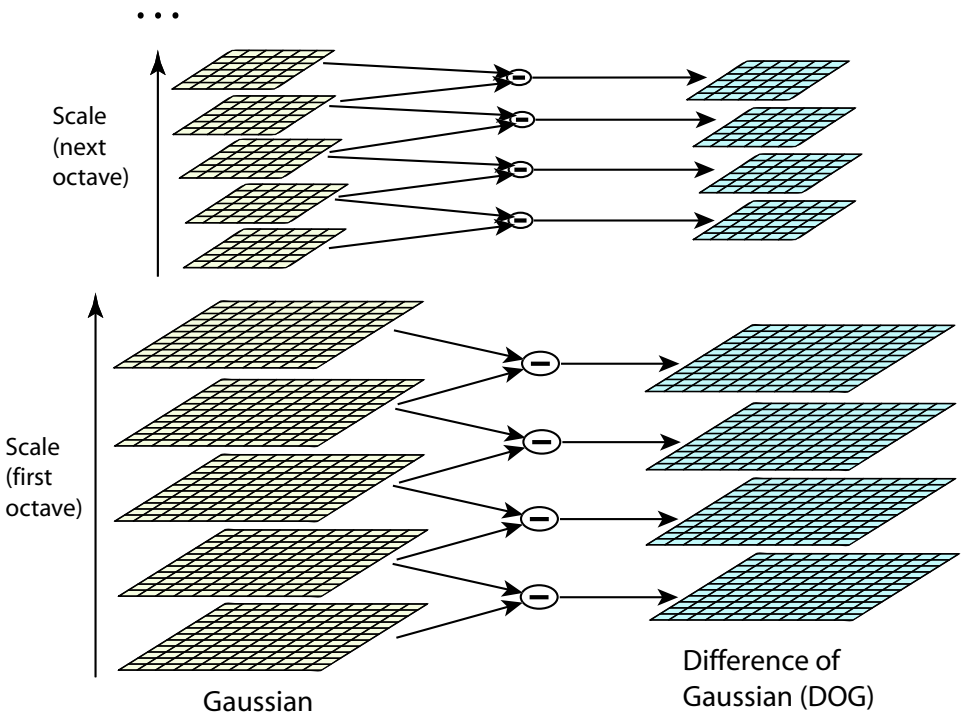


FIGURE 2.6 – Pour chaque résolution de l'espace des échelles, l'image initiale est convoluée avec une gaussienne (la partie gauche du schéma). Les images gaussiennes qui sont consécutives sont soustraites pour produire les images de différence de gaussiennes (la partie droite du schéma). Ensuite, l'image gaussienne est sous-échantillonnée par un facteur de 2, et le processus est répété. Source : [Lowe, 2004].

Afin de détecter les extrema locaux de $D(x, y, \sigma)$, chaque pixel est comparé à ses huit voisins dans l'image actuelle mais aussi avec ses neuf voisins dans l'échelle au-dessus et en-dessous (voir la figure 2.7).

Un pixel est selectionné comme extrema s'il a une valeur plus grande ou bien plus petite que tous ses voisins (voir figure 2.7).

2. Localisation des points clés : la première étape produit généralement un grand nombre d'extrema. Il faut maintenant déterminer si ces pixels extrema sont stables. Pour déterminer si les extrema sont stables, [Brown et Lowe, 2002] ont développé une méthode qui utilise la série de Taylor de la fonction $D(x, y, \sigma)$, décalé de sorte que

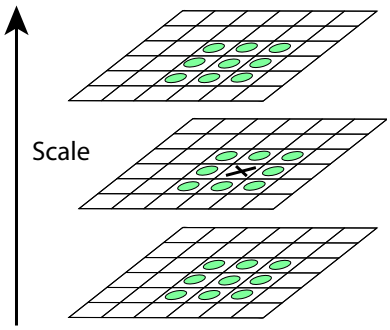


FIGURE 2.7 – Les extrema des images de différence de gaussiennes sont détectés en comparant un pixel (marqué avec un X) à ses 26 voisins dans des régions de 3 × 3 pixels à l'échelle actuelle et aux échelles adjacentes (marquées par des cercles). Source : [Lowe, 2004].

l'origine soit le point clé :

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x}, \tag{2.8}$$

où D et ses dérivés sont évalués au point d'échantillonnage et $\mathbf{x} = (x, y, \sigma)^T$ est le décalage à partir de ce point. La localisation de l'extrema $\hat{\mathbf{x}}$ est déterminée en prenant la dérivée de cette fonction par rapport à x et en le mettant à zéro, ce qui donne :

$$\hat{\mathbf{x}} = \frac{\partial^2 D^{-1}}{\partial \mathbf{x}^2}\frac{\partial D}{\partial \mathbf{x}}. \tag{2.9}$$

Enfin, pour supprimer les extrema qui se trouvent dans des zones faiblement contrastées, les auteurs de [Brown et Lowe, 2002] utilisent la fonction suivante :

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2}\frac{\partial D^T}{2\partial \mathbf{x}}\hat{\mathbf{x}}. \tag{2.10}$$

Si $D(\hat{\mathbf{x}})$ est inférieur à un seuil (0,03 dans [Brown et Lowe, 2002]), alors cet extrema est rejeté.

3. Affectation d'une orientation : le but d'affecter une orientation à chaque point clé en fonction des propriétés locales de l'image est de rendre le descripteur invariant aux rotations. L'approche pour déterminer l'orientation des points clés qui a permis d'obtenir les meilleures performances utilise le gradient [Lowe, 2004].

   À chaque position de l'image lissée $L_{x,y}$ à l'échelle du point clé, la norme du gradient, m, et l'orientation, θ, sont calculés en utilisant des différences de pixels :

$$m(x, y) = \sqrt{\big(L(x+1, y) - L(x-1, y)\big)^2 + \big(L(x, y+1) - L(x, y-1)\big)^2} \tag{2.11}$$

$$\theta(x, y) = \arctan\left(\frac{L(x, y+1) - L(x, y-1)}{L(x+1, y) - L(x-1, y)}\right) \tag{2.12}$$

Un histogramme d'orientation est construit à partir du voisinage autour du point clé. L'histogramme d'orientation est découpé en 36 intervalles de 10°. Chaque échantillon ajouté à l'histogramme est pondéré par son amplitude et par une fenêtre circulaire à pondération gaussienne avec un σ égal à 1,5 fois celui de l'échelle du point clé.

Les orientations dominantes sont représentées par les pics de l'histogramme. On détecte le pic avec la plus grande valeur maximale dans l'histogramme. Si un pic atteint au moins 80% de la plus grande valeur maximale de l'histogramme, alors on va créer un nouveau point clé à la même position et à la même échelle mais avec une direction différente. Les points clés n'atteignant pas les 80% de la valeur maximale ne sont pas gardés.

À partir de cette étape, les points clés sont définis par quatre paramètres (x, y, σ, θ), c'est-à-dire ses coordonnées cartésiennes, son facteur d'échelle et son orientation.

4. Calcul des descripteurs locaux : une fois que nous avons nos points clés, nous devons calculer leur descripteur. Pour commencer, on calcule les histogrammes des orientations relatives dans un voisinage.

   Ensuite, on calcule la norme du gradient et l'orientation de tous les pixels dans une région autour du point clé. Généralement, une fenêtre gaussienne de taille 16×16 autour du point clé est utilisée.

   Cette fenêtre est divisée en 16 blocs de 4×4. Pour chacun de ces blocs, on calcule l'orientation et la norme des gradients. Ensuite, les 16 histogrammes à 8 intervalles chacun sont concaténés et normalisés. Afin de diminuer la sensibilité du descripteur aux changements de luminosité, on plafonne généralement les valeurs à 0,2 et on normalise à nouveau l'histogramme. Nous obtenons le descripteur SIFT d'un point clé qui est un vecteur de dimension de 128.

Ce descripteur est souvent utilisé pour classifier, comme dans [Pavel *et al.*, 2009] où les auteurs font de la reconnaissance d'objets ou bien dans [Manickam *et al.*, 2018] où les auteurs font de l'identification d'empreinte digitale latente.

Le descripteur SIFT a l'avantage d'être invariant à la résolution de l'image et à l'orientation de par sa construction. En effet, même avec une rotation les points clés seront placés aux mêmes endroits dans l'image. Cependant, comme décrit dans [Azad *et al.*, 2009; Rublee *et al.*, 2011], ce descripteur est sensible au bruit et sa construction est très coûteuse en temps de calcul.

2.1.1.4 Histogrammes de gradients orientés

Le descripteur d'Histogramme de Gradient Orienté (HOG) caractérise la forme locale dans une fenêtre F en tenant compte de l'orientation de tous les pixels de la fenêtre F [Dalal et Triggs, 2005]. Il s'agit d'un descripteur dense car l'ensemble des pixels de la fenêtre F permettent d'obtenir le descripteur HOG.

Cependant l'orientation est quantifiée afin d'avoir un nombre déterminé de directions. On note N_θ le nombre de directions, et chaque angle $\theta_{(x,y)} \in]-\frac{\pi}{2}, \frac{\pi}{2}[$ est quantifié en une des N_θ directions. Généralement $N_\theta = 9$, c'est-à-dire une quantification angulaire de 20°. Plus N_θ est grand et moins l'information angulaire des pixels est quantifiée, on obtient donc une plus grande précision. De plus lorsque l'on augmente la précision, on va aussi augmenter la taille du descripteur HOG et réduire la quantification de l'information. Quand N_θ est trop grand, l'information n'est quasiment plus quantifiée et le vecteur HOG deviendra alors un vecteur parcimonieux, cela signifie qu'il contiendra beaucoup de valeurs nulles.

Chaque pixel vote pour une direction parmi les N_θ directions possibles. Pour déterminer le choix de l'orientation noté $c_{(x,y)} \in \{0, N_\theta - 1\}$ du pixel à la position (x, y), nous utilisons l'équation suivante :

$$c_{(x,y)} = \left\lfloor \left(\theta_{(x,y)} + \frac{\pi}{2}\right) \frac{N_\theta}{\pi} \right\rfloor \tag{2.13}$$

Après avoir calculé la direction $c_{(x,y)}$ choisie par le pixel à la position (x, y), on incrémente la composante correspondante dans le vecteur HOG en fonction de l'intensité locale du pixel. Une fois cette étape réalisée, il est nécessaire de normaliser le vecteur HOG noté v. Les solutions les plus utilisées pour normaliser v sont les suivantes :

— la normalisation L1, $v = \frac{v}{(||v||_1 + \epsilon)}$,

— la normalisation L2, $v = \frac{v}{(||v||_2 + \epsilon)}$,

— la normalisation L2-hys [Lowe, 2004]. La normalisation L2-hys permet de lisser les zones où le gradient dans une direction est trop important. Elle consiste à appliquer une normalisation L2, puis à tronquer ses valeurs lorsqu'elles atteignent un seuil et enfin à re-normaliser le vecteur.

Le problème de cette approche est qu'elle ne permet pas de garder l'information spatiale dans F. La solution qui est appliquée afin de ne pas perdre l'information spatiale consiste à diviser la fenêtre F en sous-blocs de tailles inférieures et de calculer l'histogramme HOG en chaque cellule. Les auteurs de [Dalal et Triggs, 2005] proposent de regrouper les sous-blocs en blocs et ensuite d'appliquer une normalisation. Ces blocs sont composés des descripteurs HOG des sous-blocs qui sont concaténés en un vecteur v. Une illustration de cette approche est présentée figure 2.8, avec en haut à gauche le découpage de l'image en blocs qui sont constitués de sous-blocs de taille 2×2. En haut à droite, nous avons l'extraction de l'histogramme de gradients orientés d'un bloc. Enfin, nous avons en bas de la figure la normalisation et la concaténation des blocs permettant d'obtenir le vecteur HOG final.

Le HOG est dépendant de la taille des sous-blocs et donc des blocs. C'est pour cette raison que les auteurs de [Qiang *et al.*, 2006] ont proposé d'utiliser des HOG en cascade. L'idée de cette approche est de construire le descripteur HOG en utilisant différentes tailles de bloc afin de calculer le vecteur HOG à des échelles et des résolutions différentes.

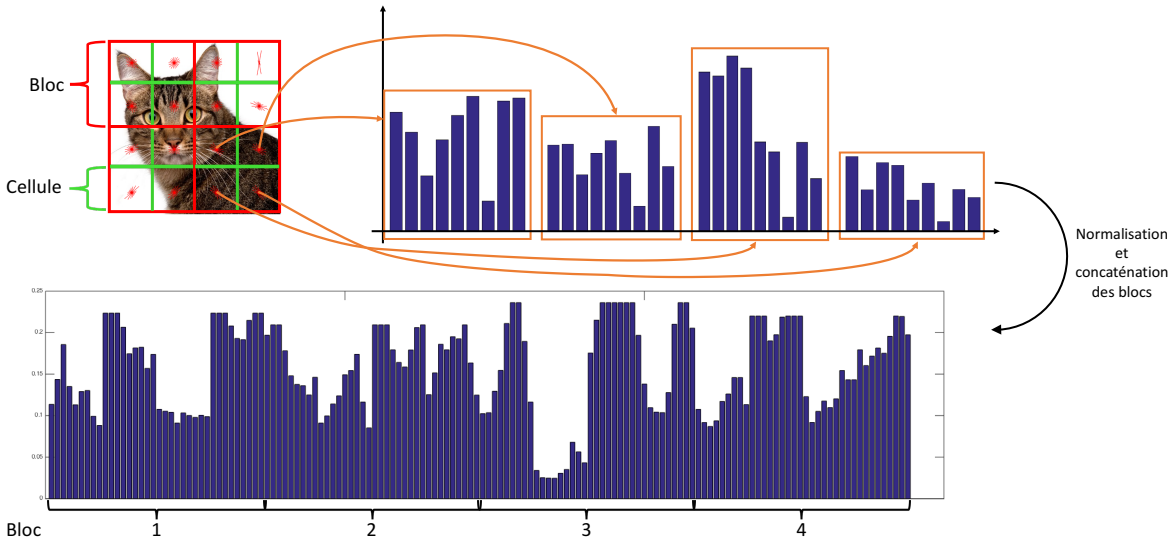


FIGURE 2.8 – Illustration de l'extraction du descripteur HOG. En haut à gauche nous avons en rouge le découpage de l'image en blocs qui sont composés de 2×2 sous-blocs représentés en vert. Ensuite on extrait l'histogramme de gradients orientés pour chaque bloc (en haut à droite du schéma). Enfin on normalise et on concatène tous les blocs pour obtenir le vecteur HOG final.

Les étapes pour calculer ce HOG-cascade sont les suivantes : 1) on fixe le nombre de sous-blocs contenus dans chaque bloc, généralement on utilise 2×2 sous-blocs par bloc, 2) avec une cellule glissante on extrait et on normalise le vecteur v de chaque bloc, 3) on fait varier horizontalement puis verticalement la taille des blocs afin de tester toutes les combinaisons de tailles possibles, puis on relance l'étape 2).

En utilisant un HOG-cascade, on augmente considérablement la représentativité mais aussi la taille du descripteur. En effet, si chaque bloc est composé de 2×2 sous-blocs, et que la taille des blocs varie de $\mathrm{B}_x^{min} \times \mathrm{B}_x^{max}$ à $\mathrm{B}_y^{min} \times \mathrm{B}_y^{max}$ avec un pas horizontal p_x et un pas vertical p_y, et si le recouvrement entre deux positions successives de la cellule glissante est de 50%, alors la taille du vecteur v notée $|v|$ sera la suivante :

$$|v| = 4\mathrm{N}_\theta \sum_{x=\mathrm{B}_x^{min}}^{\mathrm{B}_x^{max}} \sum_{y=\mathrm{B}_y^{min}}^{\mathrm{B}_y^{max}} \left\lfloor \left(4 \frac{t_\mathrm{F}^2}{x \times y} p_x p_y \right) \right\rfloor, \tag{2.14}$$

avec t_F la hauteur et la largeur de la fenêtre F. On considère dans cet exemple que la fenêtre F est carrée.

Comme on peut le constater dans [Pasquet *et al.*, 2016; Karim *et al.*, 2018], le descripteur HOG semble approprié pour la détection d'objets urbains. Cependant, comme décrit dans [Migniot, 2012], dans certains cas particuliers, ce descripteur perd en efficacité. Par exemple lorsqu'il y a des occultations ou bien un arrière plan chargé, il devient plus difficile de délimiter les objets. L'illumination peut aussi poser problème car les ombres peuvent créer des contours ne correspondant à aucun objet.

2.1.2 Algorithmes de classification

Pour effectuer une tâche de classification, de détection d'objets ou bien de segmentation sémantique dans des images, il existe plusieurs familles de classifieurs [Nasrabadi, 2007]. On peut décomposer ces classifieurs en trois grandes familles :

— les classifieurs basés sur des modèles mathématiques : ils ne nécessitent pas d'apprentissage,

— l'apprentissage automatique : les classifieurs reposent sur un apprentissage à partir de vecteurs de caractéristiques,

— le *deep learning* : les classifieurs reposent sur un apprentissage par *deep learning*.

Dans cette thèse nous nous concentrerons sur les deux familles qui nécessitent un apprentissage. Pour réaliser leur apprentissage, ces classifieurs ont besoin de bases de données d'apprentissage qui sont composées de vecteurs de caractéristiques et de labels.

L'objectif des algorithmes de classification est d'établir un modèle permettant d'étiqueter chaque donnée en l'associant à une classe aussi appelée label. Pour aboutir à ce résultat, l'apprentissage automatique classique se décompose en deux étapes. La première consiste à extraire un vecteur de caractéristiques par le biais d'un descripteur. On choisira le descripteur pour : 1) réduire la dimension de la donnée d'entrée tout en faisant en sorte que 2) les caractéristiques réagissent différemment pour chaque classe. La seconde étape consiste à apprendre à un classifieur à séparer l'espace des caractéristiques en fonction des différentes classes.

Le *deep learning* quant à lui permet à la fois de réduire la dimension des données, et de séparer l'espace des données et tout cela en une seule étape. Cette particularité qui différencie le *deep learning* de l'apprentissage automatique classique est illustrée figure 2.1.

Il existe plusieurs types d'apprentissages. Dans le cas où nous connaissons la classe de tous les objets, il s'agit d'un apprentissage supervisé. Si nous ne connaissons pas la classe de tous les objets de notre base d'apprentissage on parle d'apprentissage semi-supervisé. Enfin, lorsqu'aucun objet n'est étiqueté, on parle alors d'apprentissage non supervisé. Dans le cadre de cette thèse, nous nous sommes concentrés sur le cas de l'apprentissage supervisé.

Dans la suite du document, nous noterons $\mathbf{d}_i \in \mathbb{R}^p$ un vecteur de caractéristiques de dimension p et t_i son étiquette associée, avec $i \in \{1, \mathrm{N}\}$, et N le nombre de vecteurs dans la base d'apprentissage. Lorsque nous parlerons d'un problème binaire, c'est-à-dire un problème à deux classes, $t_i \in \{-1, 1\}$, et lorsqu'il s'agira d'un problème multi-classes, c'est-à-dire à plus de deux classes, $t_i \in \{0, \mathrm{C}_l - 1\}$, avec C_l le nombre de classes du problème.

2.1.2.1 Séparateur à Vaste Marge (SVM)

Les séparateurs à vaste marge (SVM) [Boser *et al.*, 1992; Cortes et Vapnik, 1995; Vapnik, 1995] sont à l'origine des algorithmes de classification permettant de réaliser un apprentissage supervisé, mais ils ont aussi été adaptés pour réaliser un apprentissage non

supervisé [Ben-Hur *et al.*, 2001]. Ces algorithmes sont très utilisés par la communauté de la télédétection [Huang *et al.*, 2002; Pal, 2005; Mountrakis *et al.*, 2011; Pal et Foody, 2012]. L'objectif de ces algorithmes est de trouver la meilleure séparation entre deux ensembles de données (problème binaire). Pour ce faire, ils vont chercher dans une dimension donnée le meilleur hyperplan séparateur séparant les deux jeux de données. Afin de trouver une séparation binaire le SVM calcule un vecteur appelé poids, noté $\mathbf{w}$, qui est orthogonal à l'hyperplan séparateur obtenu par la minimisation de l'expression suivante :

$$\underset{\mathbf{w}}{min}\, \mathcal{R}(\mathbf{w}) + \mathrm{C} \sum_{i=1}^{\mathrm{N}} \mathcal{L}(\mathbf{w}, \mathbf{d}_i, t_i), \tag{2.15}$$

où C est un paramètre défini par l'utilisateur qui contrôle le compromis entre $\mathcal{L}$ et $\mathcal{R}$, $\mathcal{L}$ la fonction de perte et $\mathcal{R}$ la fonction de régularisation, et $\mathbf{w}$ et $\mathbf{d}_i$ sont de dimension p. La fonction de perte permet de régler le degré d'ajustement des données en mesurant la différence entre la valeur attendue et la sortie prédite. Le choix de cette fonction est très important puisqu'il dépend du problème et va impacter les performances. La fonction de perte la plus connue est la fonction logistique [Ciocca *et al.*, 2015] qui se définit comme :

$$\mathcal{L}(\mathbf{w}, \mathbf{d}, t) = log(1 + e^{-t\mathbf{w}^{\mathrm{T}}\mathbf{d}}). \tag{2.16}$$

Les deux fonctions de régularisation les plus utilisées sont la régularisation L_1 présentée équation 2.17 et la régularisation L_2 [Fan *et al.*, 2008] présentée équation 2.18 :

$$\mathcal{R}(\mathbf{w}) = \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}, \tag{2.17}$$

$$\mathcal{R}(\mathbf{w}) = ||\mathbf{w}||. \tag{2.18}$$

Pour déterminer la classe d'appartenance d'un nouveau vecteur, il faut dans un premier temps effectuer un produit scalaire entre le vecteur poids $\mathbf{w}$ et la caractéristique du vecteur d'entrée noté $\mathbf{d}_{test}$ et ajouter un biais b :

$$s = \mathbf{w}^{\mathrm{T}} \cdot \mathbf{d}_{test} + b, \tag{2.19}$$

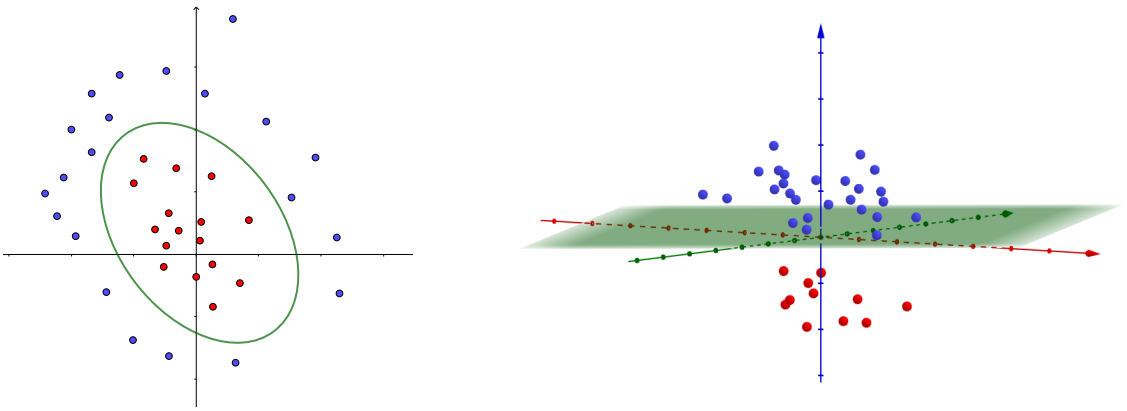
et ensuite comparer le résultat s avec un seuil pour obtenir la prédiction $\hat{t}$ du classifieur :

$$\hat{t} = \begin{cases} -1 & \text{si} \quad s < 0, \\ 1 & \text{sinon.} \end{cases} \tag{2.20}$$

Il est courant d'utiliser une fonction sigmoïde, définie comme $f(x) = \frac{1}{1+e^{-x}}$ pour normaliser le score obtenu entre 0 et 1 [Gao et Tan, 2006]. L'utilisation de cette fonction permet de connaître de quel côté est placé l'échantillon par rapport à l'hyperplan.

Cependant dans un cas réel, il est très rare de pouvoir séparer les données linéairement, c'est-à-dire dans l'espace de définition de leurs vecteurs de caractéristiques. Pour pouvoir séparer les données avec un hyperplan il est possible de transformer l'espace de

représentation des données dans une dimension supérieure. Une illustration représentant le changement de dimension permettant ensuite de séparer les données linéairement est présentée figure 2.9. Ce nouvel espace est appelé *espace de redescription*. Cette transformation d'espace est coûteuse, c'est pour cette raison que l'on utilise l'astuce du noyau (*kernel trick*).



(a) (b)

FIGURE 2.9 – Illustration du changement de dimension sur l'exemple d'une ellipse, avec à gauche le cas où les points bleus et rouges ne sont pas séparables linéairement, et à droite le résultat après le changement de dimension où les données deviennent séparables de façon linéaire.

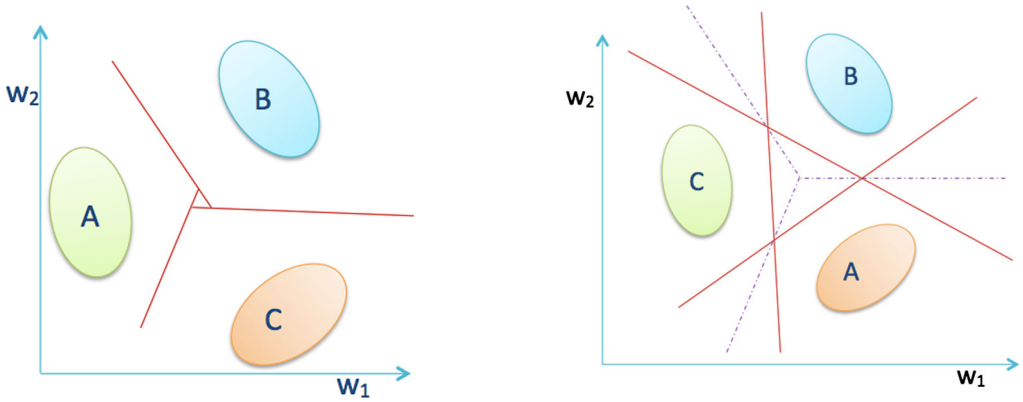
Nous allons maintenant illustrer le principe de l'astuce du noyau en prenant l'exemple de l'ellipse de la figure 2.9. Prenons par exemple $\mathbf{x} \in \mathbb{R}^2$ et $\mathbf{y} \in \mathbb{R}^2$ deux vecteurs de caractéristiques de dimension 2. Pour résoudre le cas où les données ne sont pas séparables linéairement, nous devons changer de dimension et nous faisons cela à l'aide d'une fonction. Soit ϕ une fonction de transformation permettant une redescription tel que $\phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ en utilisant la transformation suivante : $\phi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$. Faisons maintenant le produit scalaire dans l'espace de redescription :

$$\begin{aligned} \phi(\mathbf{x}).\phi(\mathbf{y}) &= (x_1^2y_1^2 + x_2^2y_2^2 + \sqrt{2}x_1x_2\sqrt{2}y_1y_2) \\ &= (x_1y_1 + x_2y_2)^2 \\ &= (x.y)^2 \end{aligned} \tag{2.21}$$

On peut constater que le calcul du produit scalaire dans ϕ peut se faire sans effectuer la transformation. L'astuce du noyau consiste à utiliser une fonction noyau K tel que $K(\mathbf{d}_i, \mathbf{d}_j) = \phi(\mathbf{d}_i).\phi(\mathbf{d}_j)$, avec $\mathbf{d}_i$ et $\mathbf{d}_j$ des vecteurs de caractéristiques. L'équation 2.21 est une fonction polynomiale de degré 2 définie par $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}.\mathbf{y})^2$. Deux des noyaux les plus utilisés avec les SVM sont le noyau polynomial : $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \cdot \mathbf{y} + 1)^k$, avec k le degré du polynôme et le noyau gaussien $K(\mathbf{x}, \mathbf{y}) = e^{-\frac{(\mathbf{x}-\mathbf{y})^2}{2\sigma^2}}$.

La complexité calculatoire des SVM est très importante, de l'ordre de $O(p.N^2)$, avec p la dimension du vecteur de caractéristiques et N la taille de la base d'apprentissage [Chang et Lin, 2011]. Les bases d'apprentissage ont un nombre d'échantillons de plus en plus im-

portant, c'est pourquoi dans la pratique cette méthode est rarement utilisée. Il est courant d'utiliser une approximation de SVM [Fan *et al.*, 2008] qui essaie de résoudre le problème sans faire de transformation en utilisant l'espace de redescription. La complexité calculatoire de cette approximation de SVM est de l'ordre de O(N), ce qui permet de traiter d'importantes bases d'apprentissage, mais ils sont aussi très efficaces pour résoudre des problème dit parcimonieux [Aldavert *et al.*, 2009], c'est-à-dire avec beaucoup de valeurs nulles.



(a) (b)

FIGURE 2.10 – Illustration des stratégies (a) *one-versus-one* et (b) *one-versus-rest*. Ce schéma illustre la différence sur la manière de traiter un problème de classification avec 3 classes. Dans le cas de l'approche *one-versus-one*, deux hyperplans sont calculés afin de séparer une classe des deux autres. Pour l'approche *one-versus-rest*, le calcul d'un seul hyperplan suffit à séparer une classe des deux autres. Source : [Chauhan *et al.*, 2018].

Les SVM ont énormément évolué au fil des années, notamment entre 1998 et 2009. Le tout premier SVM [Boser *et al.*, 1992], est appelé le SVM à marge dure. Ce SVM sépare les données à l'aide d'un hyperplan sans permettre la moindre erreur de classification. En raison de la marge dure, il n'est pas applicable aux problèmes où les données ne peuvent pas être complètement séparées par l'hyperplan. Il est donc inutilisable lorsque les données sont bruitées ou lorsqu'il y a des données aberrantes.

Le SVM de Cortes et Vapnik [Cortes et Vapnik, 1995] est une extension du SVM à marge dure, ils introduisent une variable relaxée pour permettre certaines erreurs de classification. Ainsi, il peut être appliqué à des problèmes où les données ne peuvent pas être complètement séparées par l'hyperplan. Ce SVM est principalement utilisé sur des données parcimonieuses de grandes dimension mais il est aussi très efficace sur des problèmes de classification d'images et de documents. C'est le SVM le plus populaire et le plus utilisé.

En 1998, Vapnik [Vapnik, 1998] propose la première formulation d'un SVM multi-classes qui réduit le problème multi-classes au problème binaire en considérant une classe à la fois et en la séparant du reste des classes (approche *one-versus-rest*, voir figure 2.10b). Pour C_l classes, il résout n problèmes de SVM binaires et évalue le même nombre de fonctions de décision pour classer un élément de la base de test. Il s'agit du SVM multi-classes

le plus utilisé, mais son inconvénient est qu'il est nécessaire d'entraîner plusieurs SVM et d'évaluer autant de fonctions de décision.

Un an plus tard, Weston et Watkins [Weston *et al.*, 1999] proposent le premier SVM multi-classes qui nécessite l'optimisation d'un seul SVM et l'évaluation de C_l fonctions de décision. L'idée qu'ils ont eue consiste à généraliser le problème d'optimisation des SVM binaires. Pour ce faire, ils ont construit une séparation linéaire par morceaux des C_l classes en une seule optimisation.

Dans la même année, Kreßel [Kreßel, 1999] propose une approche *one-versus-one* qui entraîne un SVM binaire pour chaque paire de classes, c'est-à-dire $\frac{C_l(C_l-1)}{2}$ SVM binaire. Il doit aussi évaluer autant de fonctions de décision. La figure 2.10 montre la différence entre l'approche *one-versus-one* et *one-versus-rest*. Comme on peut le constater sur l'exemple de la figure 2.10, pour l'approche *one-versus-one* deux hyperplans sont calculés afin de séparer une classe des deux autres. Pour l'approche *one-versus-rest* un seul hyperplan suffit à séparer une classe des deux autres.

En 2000 [Schölkopf *et al.*, 2000] propose un nouvel SVM appelé ν-SVM. Ce nouvel SVM permet de supprimer le paramètre C et introduit le paramètre ν qui donne une approximation de la fraction des vecteurs de support.

En 2005, les auteurs de [Graf *et al.*, 2005] proposent un SVM parallèle, aussi connu sous le nom de cascade de SVM. Le principe est de diviser la base d'apprentissage en sous-ensembles et d'entraîner plusieurs SVM en parallèle sur chaque sous-ensemble. une illustration du fonctionnement de ce SVM est présentée figure 2.11 où les TD_i avec $i = 1,\ldots,4$, les sous-ensembles de la base d'apprentissage.

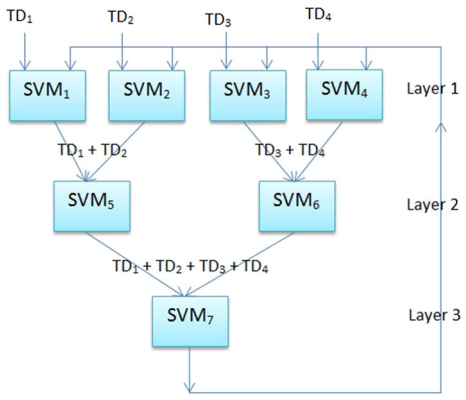


FIGURE 2.11 – Schéma du fonctionnement d'une cascade de SVM composée de 7 SVM et de 3 couches. Dans la première couche, les SVM 1,2,3 et 4 prennent chacun un sous-ensemble de la base d'entraînement, respectivement TD_1, TD_2, TD_3 et TD_4. Dans la deuxième couche, le SVM 5 prend en entrée la combinaison des résultats partiels des SVM 1 et 2, et le SVM 6 prend en entrée la combinaison des résultats partiels des SVM 3 et 4. Enfin, dans la troisième et dernière couche, le SVM 7 prend en entrée la combinaison des résultats partiels des SVM 5 et 6. Source : [Chauhan *et al.*, 2018].

Les résultats partiels (vecteurs de support) des différents SVM sont combinés en prenant les SVM deux à deux d'une même couche jusqu'à ce que l'on obtienne un seul en-

semble de vecteurs de support.

La philosophie de cette approche est de diviser le problème en sous-problèmes plus simples à résoudre, et ensuite combiner les solutions de ses sous-problèmes pour initialiser un autre SVM. Les auteurs de [Hsieh *et al.*, 2014] ont proposé une amélioration de cette approche appelée *Divide-and-Conquer SVM* (DC-SVM). Cette approche utilise le clustering de données dans l'étape de division de la base d'apprentissage de sorte que la plupart des vecteurs des sous-problèmes sont aussi les vecteurs de support du problème combiné.

En 2006, [Farquhar *et al.*, 2006] propose un SVM multi-vues. Ce type de SVM est utile dans les problèmes où les données sont obtenues à partir de différentes sources ou obtenues en utilisant différents extracteurs de caractéristiques. L'idée est d'améliorer les performances de généralisation du classifieur à travers de multiples vues du problème, par exemple en utilisant un ensemble de descripteurs ou bien des données multi-sources.

Chang et Lin [Chang et Lin, 2011] proposent en 2011 le Kernel SVM. Il est utilisé lorsque les données ne sont pas linéairement séparables où lorsqu'il n'est pas possible de trouver un hyperplan séparable dans l'espace d'entrée. Tout problème SVM convexe peut être converti en Kernel SVM correspondant.

Pour gérer le problème des données bruitées et aberrantes, comme par exemple une donnée qui n'a pas le bon label dans une base d'apprentissage, [Nie *et al.*, 2017] propose le *CappedSVM*. Ces données aberrantes peuvent induire en erreur les classifieurs durant l'apprentissage, ainsi, les classifieurs appris ne sont pas optimaux et leurs performances en sont réduites. Le CappedSVM utilise une nouvelle formulation du problème en utilisant une fonction de perte basée sur une norme l_p bornée.

2.1.2.2 Les Random Forests

Les forêts aléatoires, ou *Random Forests* (RF) [Ho, 1995, 1998; Breiman, 2001; Pal, 2005; Cutler *et al.*, 2007; Belgiu et Drăguţ, 2016; He *et al.*, 2017b], sont des Ensembles de Classifieurs, cela signifie qu'elles sont composées de plusieurs classifieurs de faible complexité. Les RF sont composées d'une multitude d'arbres de décision [Quinlan, 1986; Utgoff, 1989, 1994] qui sont entraînés sur des sous-ensembles de la base d'apprentissage. La figure 2.12 illustre le fonctionnement d'un arbre de décision. On peut aussi voir chaque arbre de décision comme une décomposition du problème de classification en une suite de tests correspondant à une partition de l'espace des données en sous-régions homogènes selon l'ensemble de variables discriminantes. Dans l'exemple de la figure 2.12, l'ensemble de variables discriminantes sont la couleur, la taille et la forme. Pour classer une nouvelle entrée, on part de la racine et on effectue les tests des nœuds jusqu'à ce qu'une feuille soit atteinte.

Le fonctionnement d'un algorithme de forêt aléatoire est le suivant : nous notons $\mathbb{Z} = \{(\mathbf{d}_i, t_i)\}_{i=1}^{i=\mathrm{N}}$ la base d'apprentissage composée de vecteurs de caractéristiques $\mathbf{d}_i \in \mathbb{R}^p$ et $t_i \in \{0, \mathrm{C}_l - 1\}$ les étiquettes associées, et $i \in \{1, \mathrm{N}\}$ où N est le nombre d'éléments de la base d'apprentissage. Pour la construction d'un arbre a_b de la forêt, avec $b \in [1, \ldots, \mathrm{B}]$ et

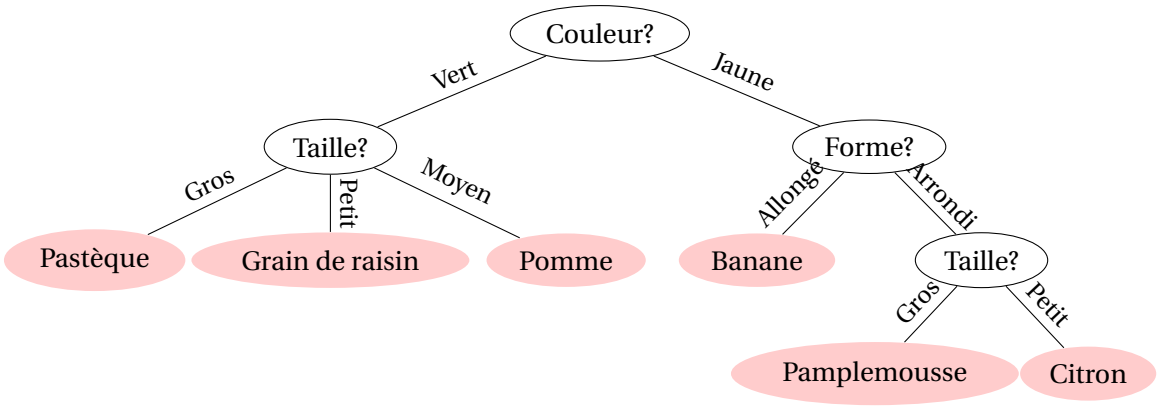


FIGURE 2.12 – Exemple d'arbre de décision.

B le nombre d'arbres de la forêt, on tire aléatoirement des éléments de $\mathbb{Z}$ pour créer une sous-base d'apprentissage $\mathbb{Z}_b$ (de taille inférieure à N) avec remise dans $\mathbb{Z}$. On construit un arbre de décision a_b avec $b \in [1,\ldots,\mathrm{B}]$ à partir de la sous-base d'apprentissage $\mathbb{Z}_b$. Pour estimer la classe d'un échantillon $\mathbf{d}_{test}$ lors de la phase de test, on peut par exemple utiliser la moyenne des prédictions de tous les arbres sur le vecteur $\mathbf{d}_{test}$:

$$\hat{t} = \frac{1}{\mathrm{B}} \sum_{b=1}^{\mathrm{B}} a_b(\mathbf{d}_{test}), \tag{2.22}$$

avec $a_b : \begin{array}{|ccc} \mathbb{R}^p & \rightarrow & \mathbb{R} \\ \mathbf{d}_{test} & \mapsto & \{0,\ldots,\mathrm{C}_l - 1\} \end{array}$ la prédiction d'un arbre a_b.

Le seul paramètre à définir pour ce classifieur est B. Suivant la nature et la taille de la base d'apprentissage, le nombre d'arbres utilisés peut être de plusieurs centaines à plusieurs milliers. La valeur optimale d'arbres à utiliser peut être déterminée par validation croisée [James *et al.*, 2013]. Cela signifie que l'on va utiliser une base de validation ne faisant pas partie de la base d'apprentissage ni de la base de test pour fixer ce paramètre.

Chaque arbre va être construit à partir d'un sous-échantillon des vecteurs de caractéristiques, chaque arbre apprend donc de manière indépendante. Cette stratégie permet d'utiliser le hasard pour améliorer les performances d'algorithmes de "faibles" performances. Ce type de méthode s'appelle le *bagging* [Breiman, 1996]. L'idée est d'utiliser une combinaison ou une agrégation d'un grand nombre de modèles tout en évitant le phénomène de sur-apprentissage. Durant l'apprentissage, le but est d'obtenir des arbres les plus décorrélés possible.

Pour améliorer les performances, la méthode du sous-espace aléatoire [Ho, 1998] (*Random Subspace Method*, ou RSM en anglais) est utilisée. Cette méthode consiste à sous-échantillonner les données d'apprentissage. Les échantillons $\mathbf{d}_i$, sont des vecteurs de caractéristiques de dimension p. La méthode RSM consiste à sélectionner aléatoirement r composantes parmi les p composantes des vecteurs $\mathbf{d}_i$, avec $r << p$. Nous obtenons une nouvelle base d'apprentissage $\tilde{\mathbb{Z}} = ((\tilde{\mathbf{d}}_1, t_1), (\tilde{\mathbf{d}}_2, t_2), \ldots, (\tilde{\mathbf{d}}_n, t_n))$ composée de n vecteurs de caractéristiques de dimension r. C'est sur cette base d'apprentissage que les arbres de décision vont être entraînés.

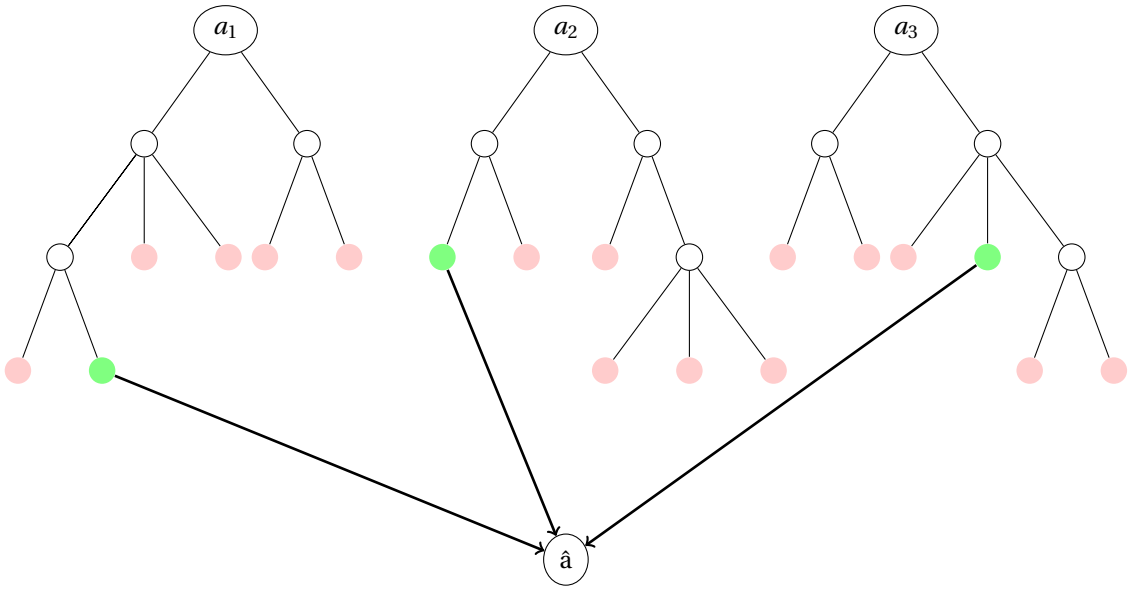


FIGURE 2.13 – Exemple de forêt aléatoire composée de 3 arbres de décision a_1, a_2 et a_3.

L'avantage des forêts aléatoires est qu'elles sont multi-classes par nature et faciles à interpréter.

Algorithme 2 : Algorithme d'apprentissage d'une forêt aléatoire

Données : $\mathbb{Z} = \{(\mathbf{d}_1, t_1), \ldots, (\mathbf{d}_n, t_n)\}_{i=1}^{i=\mathrm{N}}$
Parameters : B le nombre d'arbres, r le nombre de caractéristiques à sélectionner
Résultat : Une forêt composée de B arbres a_b qui sont des prédicteurs

pour $b = 1, \cdots, \mathrm{B}$ **faire**
  Rééchantillonnage aléatoire avec remise de $\mathbb{Z}$ pour obtenir $\mathbb{Z}_b$;
  $\widetilde{\mathbb{Z}}_b \leftarrow \varnothing$;
  pour $\boldsymbol{d}_i \in \mathbb{Z}_b$ **faire**
    $\tilde{\mathbf{d}}_i \leftarrow \mathrm{RSM}(\mathbf{d}_i, r)$;
    $\widetilde{\mathbb{Z}}_b \leftarrow \widetilde{\mathbb{Z}}_b \cup \{\tilde{\mathbf{d}}_i\}$;
  fin
  Apprentissage de a_b sur $\widetilde{\mathbb{Z}}_b$;
fin

2.1.3 Bilan de l'apprentissage automatique en deux étapes

Nous avons présenté dans cette partie différentes stratégies que propose l'apprentissage automatique en deux étapes afin de réaliser une tâche de classification. L'inconvénient de cette approche est qu'elle nécessite une étape d'extraction des vecteurs de caractéristiques. En effet, comme nous avons pu le voir, les résultats obtenus en utilisant un classifieur dépendent du ou des descripteurs que l'on va utiliser pour entraîner le classifieur. De plus, les descripteurs utilisés pour la détection d'un objet en particulier peuvent

être totalement inefficaces dès que l'on se focalise sur un autre objet, cette approche est donc difficilement généralisable.

Depuis 2012, une méthode permettant de réaliser une tâche de classification sans avoir à extraire de vecteurs de caractéristiques est de plus en plus utilisée. Il s'agit des réseaux de neurones et plus particulièrement des réseaux de neurones convolutionnels (CNN) lorsque les données traitées sont des images. Cette méthode réalise l'extraction des caractéristiques et la classification via une optimisation jointe, l'utilisation d'un descripteur n'est donc plus nécessaire. Nous présentons cette approche ci-dessous.

2.2 Les réseaux de neurones convolutionnels

2.2.1 Architecture d'un réseau de neurones classiques et rétropropagation

Nous avons vu que les méthodes d'apprentissage automatique supervisées classiques se déroulent en deux étapes. La première consiste à extraire des caractéristiques discriminantes qui ont été définies par l'utilisateur. La deuxième phase va utiliser un classifieur afin d'apprendre la meilleure séparation entre les différentes classes. Cependant les caractéristiques extraites lors de la première phase ne sont pas forcément optimales pour la classification que l'on souhaite accomplir. Cette phase d'extraction des caractéristiques est difficilement généralisable d'un problème de classification à un autre. Afin de s'affranchir de cette étape, une solution consiste à utiliser des réseaux de neurones profonds. Ce type d'apprentissage permet de réaliser la phase d'extraction des caractéristiques et de classification de manière jointe. Un réseau de neurones est un modèle mathématique dont la conception est inspirée des neurones biologiques. Les représentations de ces réseaux sont très fortement inspirées de l'article [Lettvin *et al.*, 1959].

Les réseaux de neurones profonds ont connu un essor important dans les années 2000 grâce à leurs records dans différentes compétitions. Ils ont notamment gagné la compétition *ImageNet Large Scale Visual Recognition Competition* (ILSVRC) en 2012, 2013, 2014, 2015, 2016 et 2017 [Krizhevský *et al.*, 2012; Zeiler et Fergus, 2013; Simonyan et Zisserman, 2015; Szegedy *et al.*, 2015; Kaiming *et al.*, 2016; Hu *et al.*, 2017].

Un réseau est représenté sous la forme d'un graphe et est organisé en couches de neurones. Chaque couche prend en entrée une ou plusieurs couches de rang inférieur. La figure 2.14 représente un réseau de neurones. Nous noterons $p_k^{(c)}$ le neurone k de la couche c, avec $k \in \{1, \mathrm{N}^{(c)}\}$ et $\mathrm{N}^{(c)}$ le nombre de neurones de la couche c. Si la couche c est connectée à la couche $c-1$, alors cette couche est dite *cachée* (la couche bleue de la figure). Si la couche c n'est pas connectée à une couche de rang inférieur, alors c est une couche *d'entrée* (la couche verte de la figure) et prend en entrée les données qui sont passées au réseau. Enfin, si une couche c n'est pas connectée à une couche de rang supérieur, alors il s'agit d'une couche de *sortie* (la couche blanche de la figure).

Un réseau peut s'organiser de deux façons différentes : soit de façon acyclique, soit

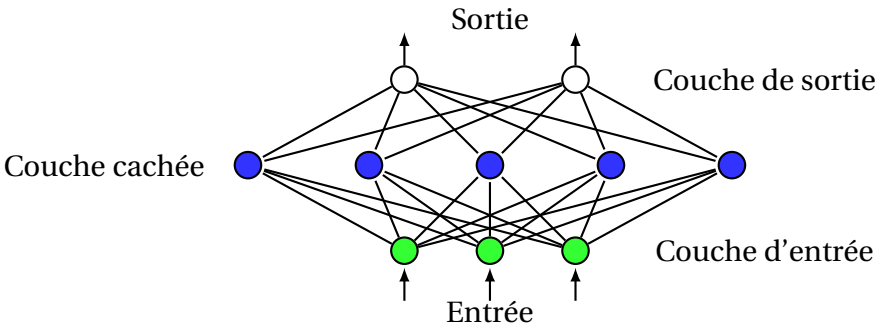


FIGURE 2.14 – Structure d'un réseau de neurones

de façon cyclique. Les réseaux cycliques sont appelés des réseaux récurrents [Rumelhart *et al.*, 1986] et leurs couches se connectent en formant au moins un cycle. Les réseaux acycliques sont les réseaux les plus courants et le résultat d'une couche au rang r est uniquement transmis à une ou plusieurs couches de rang $(r+l)$ avec $l \in \mathbb{R}_+^*$.

La figure 2.15 représente le fonctionnement d'un neurone avec $x_i^{k^{(c)}} \in \mathbb{R}$ et $w_i^{k^{(c)}} \in \mathbb{R}$ les composantes du vecteur d'entrée $\mathbf{x}_k^{(c)}$ et du vecteur poids $\mathbf{w}_k^{(c)}$, $i \in \{1,\dots,n\}$ le nombre de composantes et $b_k^{(c)} \in \mathbb{R}$ le biais du neurone $p_k^{(c)}$. Dans un premier temps, un neurone réalise le produit scalaire entre le vecteur d'entrée $\mathbf{x}_k^{(c)} \in \mathbb{R}^n$ et le vecteur poids $\mathbf{w}_k^{(c)} \in \mathbb{R}^n$ auquel nous ajoutons le biais $b_k^{(c)}$ permettant de réguler le seuil d'activation du neurone, ce résultat est noté $o_k^{(c)}$. Dans un second temps, nous appliquons une fonction f appelée fonction d'activation sur $o_k^{(c)} \in \mathbb{R}$, le résultat est noté $s_k^{(c)} \in \mathbb{R}$. Afin que le réseau puisse résoudre des problèmes complexes, il est nécessaire que la fonction d'activation f soit non linéaire.

$$o_k^{(c)} = \mathbf{w}_k^{(c)} \cdot \mathbf{x}_k^{(c)} + b_k^{(c)} \tag{2.23}$$

$$s_k^{(c)} = f(o_k^{(c)}) \tag{2.24}$$

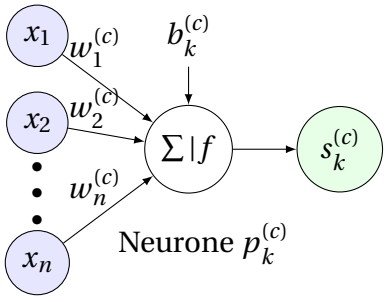


FIGURE 2.15 – Représentation d'un neurone $p_k^{(c)}$ de la couche c.

À l'initialisation du réseau, les poids des neurones sont initialisés aléatoirement en suivant une distribution donnée. Les types de distributions les plus utilisées sont la distribution gaussienne, uniforme, ou bien Xavier [Glorot et Bengio, 2010]. La loi Xavier est une initialisation uniforme où chaque couche va calculer les bornes maximale et minimale optimales en tenant compte du nombre de connexions entrantes n_e et sortantes n_s

d'un neurone. Les poids de ces neurones seront initialisés en suivant la loi $\mathrm{U}\left[-\sqrt{\frac{6}{n_e+n_s}}, \sqrt{\frac{6}{n_e+n_s}}\right]$ avec U une distribution uniforme.

La couche de sortie possède autant de neurones qu'il y a de classes à discriminer dans la base d'apprentissage. Chaque neurone de sortie est donc associé à une classe et retourne le score d'appartenance à sa classe. Pour connaître le score d'appartenance d'un objet à une certaine classe, il est intéressant de normaliser les valeurs de sortie dans l'intervalle [0,1]. La fonction qui est classiquement utilisée est une fonction *softmax* qui est définie comme suit :

$$s_k = \frac{exp(o_k^{(c_s)})}{\sum_{j=1}^{\mathrm{N}(c_s)} exp(o_j^{(c_s)})}, \tag{2.25}$$

avec $o_k^{(c_s)}$ et $o_j^{(c_s)}$ les sorties des neurones k et j sans fonction d'activation. Il est usuel d'appeler le résultat de la sortie de chaque classe la **probabilité** d'appartenance à la classe.

Une fois que tous les paramètres du réseau de neurones sont initialisés, nous pouvons effectuer la phase d'apprentissage. Pour ce faire, nous utilisons une méthode appelée rétropropagation [Rumelhart *et al.*, 1986]. La rétropropagation est une méthode itérative pour calculer le gradient de l'erreur pour chaque neurone d'un réseau de neurones, de la dernière couche vers la première.

Lors de l'activation du réseau, chaque neurone du réseau va "transmettre" une valeur aux neurones qui lui sont connectés. Le réseau est donc "activé" en lui donnant en entrée un vecteur (ie. une image) et un vecteur $\mathbf{t} = (t_0, \ldots, t_{\mathrm{C}_l-1})$ dont la $\mathrm{j}^{ème}$ composante vaut 1 si la classe de l'image en entrée est la $\mathrm{j}^{ème}$, et toutes les autres sont à 0. Le but de l'algorithme de rétropropagation est de minimiser l'erreur quadratique E (définie équation 2.26) du réseau en utilisant la descente de gradient. On souhaite donc que la sortie du neurone $s_j^{(c_s)}$ vale 1 et que les autres sorties valent 0.

$$\mathrm{E} = \frac{1}{2}\sum_{i=1}^{\mathrm{N}^{(c)}} (t_i - s_i^{(c_s)})^2, \tag{2.26}$$

Afin de simplifier les notations nous allons reprendre les notations de la figure 2.16. Notons $w_{ij} \in \mathbb{R}$ le poids reliant le neurone $p_i^{c_1}$ au neurone $p_j^{c_2}$ et c_2 la couche qui prend en entrée la couche c_1. L'objectif de la rétropropagation est de corriger tous les poids w_{ij} du réseau avec une valeur de correction Δw_{ij}. Cette valeur de correction est choisie de façon à ce que le poids corrigé minimise l'erreur E, elle suit donc la loi $\Delta w_{ij} \sim -\frac{\delta \mathrm{E}}{\delta w_{ij}}$. L'idée est donc de mettre à jour w_{ij} en le modifiant d'une faible valeur proportionnelle à l'opposé du gradient. En utilisant le théorème de dérivation des fonctions composées, on peut écrire le gradient $\frac{\partial \mathrm{E}}{\partial w_{ij}}$ de la manière suivante :

$$\frac{\partial \mathrm{E}}{\partial w_{ij}} = \underbrace{\frac{\partial \mathrm{E}}{\partial s_j^{(c_2)}}}_{3} \underbrace{\frac{\partial s_j^{(c_2)}}{\partial o_j^{(c_2)}}}_{2} \underbrace{\frac{\partial o_j^{(c_2)}}{\partial w_{ij}}}_{1}, \tag{2.27}$$

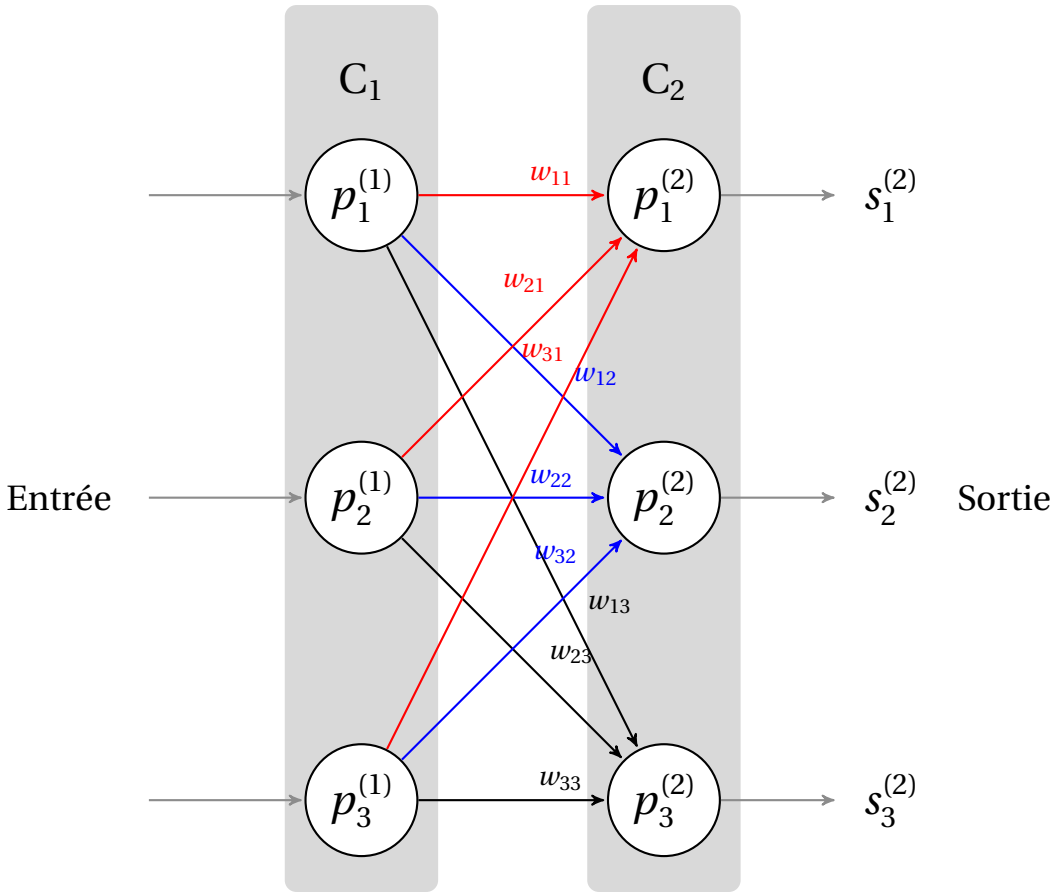


FIGURE 2.16 – Schéma représentant un réseau à deux couches.

avec $o_j^{(c_2)} = \mathbf{w}_j^{(c_2)} \cdot \mathbf{x}_j^{(c_2)} + b_j^{(c_2)}$.

Détaillons les différents termes, commençons par le terme 1 :

$$\frac{\partial o_j^{(c_2)}}{\partial w_{ij}} = \frac{\partial(\mathbf{w}_j^{(c_2)} \cdot \mathbf{x}_j^{(c_2)} + b_j^{(c_2)})}{\partial w_{ij}} = \frac{\partial(\sum_{k=1}^{N^{(c_1)}} w_{kj} \cdot s_k^{(c_1)})}{\partial w_{ij}} = s_i^{(c_1)} \tag{2.28}$$

avec $s_k^{(c_1)} = f(o_k^{(c_1)})$.

En détaillant le terme 2, nous obtenons :

$$\frac{\partial s_j^{(c_2)}}{\partial o_j^{(c_2)}} = \frac{\partial(f(o_j^{(c_2)})}{\partial o_j^{(c_2)}} = f'(o_j^{(c_2)}) \tag{2.29}$$

f' est la dérivée de la fonction d'activation. Elle est appelée fonction de transfert et elle détermine la force de la correction de l'erreur.

Enfin, pour le terme 3, il faut différencier deux cas, lorsque le neurone appartient à une couche de sortie ou à une couche cachée. Si le neurone appartient à une couche de sortie :

$$\frac{\partial \mathrm{E}}{\partial s_j^{(c_2)}} = \frac{\partial \frac{1}{2} \sum_{i=1}^{N^{(c_2)}} (t_i - s_i^{(c_2)})^2}{\partial s_j^{(c_2)}} = -(t_j - s_j^{(c_2)}) \tag{2.30}$$

Si le neurone appartient à une couche cachée :

$$\frac{\partial \mathrm{E}}{\partial s_j^{(c_2)}} = \sum_{c \in \mathrm{F}(c_2)} \sum_{k=1}^{\mathrm{N}^{(c)}} \frac{\partial \mathrm{E}}{\partial o_k^{(c)}} \frac{\partial o_k^{(c)}}{\partial s_j^{(c_2)}} \tag{2.31}$$

Avec $\mathrm{F}(c_2)$ les couches filles de la couche c_2, c'est-à-dire les couches prenant en entrée la sortie de c_2, puisque l'on itère de la couche de sortie vers la couche d'entrée du réseau.

De plus, comme nous avons $\frac{\partial o_k^{(c)}}{\partial s_j^{(c_2)}} = \frac{\partial (\mathbf{w}_k^c \cdot \mathbf{x}_k^c + \mathbf{b}_k^c)}{\partial s_j^{(c_2)}} = w_{kj}$, nous obtenons :

$$\frac{\partial \mathrm{E}}{\partial o_k^{(c)}} \frac{\partial o_k^{(c)}}{\partial s_j^{(c_2)}} = \frac{\partial \mathrm{E}}{\partial s_k^{(c)}} \frac{\partial s_k^{(c)}}{\partial o_k^{(c)}} w_{kj} \tag{2.32}$$

On reconnaît ici, les termes 2 et 3 de l'équation de départ. On note $\delta_k^{(c)}$ l'erreur du neurone $p_k^{(c)}$, tel que $\delta_k^{(c)} = \frac{\partial \mathrm{E}}{\partial s_k^{(c)}} \frac{\partial s_k^{(c)}}{\partial o_k^{(c)}}$.

En regroupant les termes, on obtient l'équation suivante :

$$\frac{\partial \mathrm{E}}{\partial w_{ij}} = \underbrace{f'(o_j^{(c_2)})}_{2} \times \underbrace{s_i^{(c_1)}}_{1} \times \begin{cases} \underbrace{(t_j - s_j^{(c_2)})}_{3} & \text{si } c_2 \text{ est une couche de sortie} \\ \underbrace{\left(\sum_{c \in \mathrm{F}(c_2)} \sum_{k=1}^{\mathrm{N}^{(c)}} \delta_k^{(c)} w_{kj} \right)}_{3} & \text{sinon.} \end{cases} \tag{2.33}$$

Une fois que nous avons calculé $\frac{\partial \mathrm{E}}{\partial w_{ij}}$, nous pouvons calculer le facteur de correction Δw_{ij} que nous devons appliquer sur w_{ij}. Pour éviter d'avoir des variations de poids trop importantes et brusques pouvant être causées par des points aberrants, le facteur de correction Δw_{ij} est multiplié par ce que l'on appelle un coefficient d'apprentissage. Ce coefficient d'apprentissage noté $\epsilon \in [0,1]$, va diminuer au cours de l'apprentissage pour permettre une convergence.

$$\Delta w_{ij} = \Delta w_{ij} - \epsilon \frac{\partial \mathrm{E}}{\partial w_{ij}} \tag{2.34}$$

Le problème de cette méthode c'est qu'elle permet de trouver un minimum local mais pas forcément le minimum global. Pour empêcher ce phénomène, il est courant d'ajouter un terme d'inertie noté $\lambda \in [0,1]$, qui va mémoriser la direction et la force du gradient par rapport à la mise à jour à effectuer. Le facteur de correction prenant en compte le coefficient d'apprentissage ϵ et le moment d'inertie λ se définissent comme suit :

$$\Delta w_{ij} = \lambda \Delta w_{ij} - \epsilon \frac{\partial \mathrm{E}}{\partial w_{ij}} \tag{2.35}$$

Durant l'apprentissage, les poids vont donc se mettre à jour de la manière suivante :

$$w_{ij} = w_{ij} + \Delta w_{ij} \tag{2.36}$$

Dans le cas présenté précédemment on met à jour le réseau de neurones après avoir présenté un élément de la base d'apprentissage, c'est-à-dire après chaque itération (une itération correspond au traitement par le réseau d'un élément de la base d'apprentissage). Ce type d'apprentissage s'appelle un apprentissage en ligne (*on-line*). Pour que le réseau puisse converger, il faut effectuer plusieurs passes sur la totalité de la base d'apprentissage, ce qui représente un coût calculatoire important. Un autre problème lié à ce type d'apprentissage est que les fluctuations provoquées par la diversité des échantillons peuvent aboutir à des mises à jours importantes des poids, ce qui peut ralentir, voire empêcher, un réseau de converger.

Pour éviter ce problème de fluctuation, l'utilisation d'un apprentissage dit stochastique est préféré. Cette approche va calculer une seule correction des poids pour la totalité de la base d'apprentissage en n'effectuant qu'une seule mise à jour en considérant la correction moyenne de tous les poids [Saad, 1998]. Ce type de descente de gradient permet de lisser l'erreur sur l'ensemble de la base d'apprentissage, ce qui réduit les fluctuations et facilite la convergence. Le problème de cette approche est lorsque l'on utilise une base d'apprentissage de grande dimension. En effet, parcourir l'ensemble de la base d'apprentissage pour réaliser une mise à jour devient trop coûteux.

Une solution intermédiaire consiste à utiliser des mises à jour par paquets. L'idée est de calculer l'erreur et de faire la mise à jour des poids sur un paquet composé de k éléments de la base d'apprentissage. Cette approche nous permet de lisser l'erreur sur des paquets de petites tailles ce qui est moins coûteux que de le faire sur la totalité de la base. Un autre avantage de cette méthode, c'est qu'elle est facilement parallélisable sur GPU puisque nous effectuons k fois la même opération. Lorsque $k = 1$ nous sommes dans le cas de l'apprentissage en ligne, et lorsque k est égal au nombre d'éléments de la base d'apprentissage alors nous sommes dans le cas de l'apprentissage stochastique.

De nouvelles méthodes utilisant des mises à jour par paquets permettant de réduire le temps de convergence ont vu le jour. Par exemple AdaGrad [Duchi *et al.*, 2011] qui est un algorithme pour l'optimisation basée sur le gradient qui adapte le coefficient d'apprentissage appliqué sur les paramètres du réseau. Il effectue des mises à jour plus importantes lorsqu'il y a des caractéristiques peu fréquentes et plus petites pour les caractéristiques fréquentes. Cela veut dire que Adagrad va modifier le coefficient d'apprentissage général d'un paramètre (w ou b) en fonction des gradients calculés jusque là. Pour cette raison, il est bien adapté pour traiter des données parcimonieuses, c'est-à-dire des données qui contiennent beaucoup de valeurs nulles. Dean et al. [Dean *et al.*, 2012] ont montré qu'Adagrad améliorait grandement les performances comparé à une mise à jour des poids classique.

Des extensions d'AdaGrad telles que Adadelta [Zeiler, 2012] et Adam [Kingma et Ba, 2014] (*Adaptive Moment Estimation*) sont apparues en 2012 et 2014. Ces extensions proposent de nouvelles méthodes permettant de stocker le gradient calculé lors de la rétropropagation et ainsi de réduire encore plus le temps de convergence.

Lors de la phase d'apprentissage, l'utilisation d'une base de validation est indispen-

sable. Cette base est constituée d'éléments de la base d'entraînement qui ne sont pas utilisés lors de l'apprentissage. Le rôle de la base d'évaluation est dans un premier temps d'évaluer les performances du modèle, mais aussi d'arrêter la phase d'apprentissage lorsque le réseau a convergé. On considère que le point de convergence est atteint lorsque l'erreur ne diminue plus sur la base de validation au court du temps et fluctue. Si la phase d'apprentissage n'est pas arrêtée lorsque ce point est atteint, un phénomène de sur-apprentissage peut apparaître. Ce phénomène se caractérise par une erreur nulle sur la base d'entraînement et une augmentation de l'erreur sur la base de validation. On parle de sur-apprentissage lorsque le réseau apprend tous les cas spécifiques de la base d'apprentissage sans généraliser les concepts, on dit aussi que le réseau a appris "par cœur" les données.

2.2.2 Comment définir les fonctions d'activation

Comme cela a été expliqué précédemment, la fonction d'activation f est fondamentale pour résoudre la non linéarité, voir équation 2.24. À l'origine, la première fonction non linéaire utilisée comparait la sortie d'un neurone (voir équation 2.23) à un seuil [McCulloch et Pitts, 1943]. Cette fonction binaire appelée fonction Heaviside, ou marche d'escalier, est de la forme :

$$f(x) = \begin{cases} 0 & \text{si} \quad x < 0 \\ 1 & \text{sinon} \end{cases} \tag{2.37}$$

Le problème de cette fonction c'est que la binarisation du signal entraîne une perte d'information qui peut être grande. Par la suite, d'autres fonctions ont été proposées où la transformation du signal est bornée. On retrouve parmi les fonctions les plus utilisées les fonctions de type tangente hyperbolique : $f(x) = \frac{2}{1+e^{-2x}} - 1$, sigmoïde : $f(x) = \frac{1}{1+e^{-x}}$, softsign : $f(x) = \frac{x}{1+|x|}$. En 2010 [Glorot et Bengio, 2010] comparent les 3 fonctions précédentes et montrent que les fonctions softsign et tangente hyperbolique obtiennent des performances semblables. Cependant, le calcul de la dérivée de ces fonctions pour réaliser la rétropropagation est très "coûteux".

Durant cette même année [Nair et Hinton, 2010] propose l'utilisation d'une nouvelle fonction d'activation, la fonction *Rectified Linear Units* ou ReLU. Cette fonction permet de réduire les coûts calculatoires mais réduit aussi le temps de convergence. Cette fonction est définie comme : $f(x) = max(0, x)$, et sa fonction de transfert comme :

$$f'(x) = \begin{cases} 0 & \text{si} \quad x < 0 \\ 1 & \text{sinon.} \end{cases} \tag{2.38}$$

Cette fonction est encore très utilisée aujourd'hui. En effet [Krizhevský *et al.*, 2012] ont montré que le ReLU converge 6 fois plus rapidement que avec la tangente hyperbolique. La vitesse de convergence lors de l'utilisation du ReLU est dûe à sa fonction de transfert simple, soit elle transmet une erreur nulle, soit l'erreur dans sa totalité. Cette fonction

permet d'éviter le phénomène d'atténuation du gradient contrairement à la tangente hyperbolique par exemple.

Depuis l'apparition de cette fonction d'activation, de nombreuses fonctions basées sur le ReLU ont vu le jour. On peut noter que ces nouvelles fonctions utilisent généralement un paramètre qui est alors appris lors de l'apprentissage, comme par exemple :

— la fonction *Parametric Rectified Linear Units* ou PReLU [He *et al.*, 2015] :

$$f(x) = \begin{cases} \alpha x & \text{Si} \quad x < 0 \\ x & \text{sinon,} \end{cases} \tag{2.39}$$

— ou la fonction *Exponential Linear Units* ou ELU [Clevert *et al.*, 2015] :

$$f(x) = \begin{cases} \alpha(e^x - 1) & \text{Si} \quad x < 0 \\ x & \text{sinon.} \end{cases} \tag{2.40}$$

Les auteurs de [Qian *et al.*, 2018] combinent de manière différente les fonctions de la famille des *Rectified Units* pour créer des fonctions d'activation adaptatives permettant d'avoir une réduction de 1,35% de l'erreur avec le réseau GoogleNet [Szegedy *et al.*, 2015] sur la base ImageNet.

2.2.3 Intégration de convolutions dans un réseau de neurones

Les réseaux de neurones convolutionnels (CNNs) [Fukushima, 1975; Fukushima et Miyake, 1982; LeCun *et al.*, 1995] sont des architectures très adaptées dans le domaine de la reconnaissance dans les images ou bien les vidéos, ce sont des outils très puissants dans les problèmes de vision.

La particularité des CNNs est qu'ils utilisent des neurones de type convolutif. Ce type de neurone applique une opération de convolution, c'est-à-dire un filtre que l'on appelle noyau, sur son signal d'entrée.

Traitons le cas général d'une convolution 2D. Considérons un noyau $\mathrm{K} \in \{0 \dots k_1\} \times \{0 \dots k_2\}$ et une image $\mathrm{I} \in \mathbb{R}^{w \times h}$, la convolution de I par K, notée $\mathrm{I} * \mathrm{K}$, est définie comme :

$$(\mathrm{I} * \mathrm{K})_{(i,j)} = \sum_{i'=0}^{k_1} \sum_{j'=0}^{k_2} \mathrm{I}_{(i+i'-\lfloor \frac{k_1}{2} \rfloor, j+j'-\lfloor \frac{k_2}{2} \rfloor)} \cdot \mathrm{K}_{(i',j')} \tag{2.41}$$

Dans le cas d'une image multi-canaux, nous avons la largeur, la hauteur et le nombre de canaux à prendre en compte. Pour chaque neurone $p_i^{(c)}$, le noyau, ou le poids associé est noté $\mathbf{w}_k^{(c)} \in \mathbb{R}^{k_1 \times k_2 \times c}$ avec k_1 et k_2 la hauteur et la largeur du noyau et c le nombre de canaux. Les deux premières dimensions du noyau représentent la largeur et la hauteur de la matrice de convolution et la troisième dimension représente le canal sur lequel la convolution est appliquée. Nous notons $\mathbf{w}_k^{(c)}(p)$ le noyau 2D appliqué sur le canal p.

Comme pour un réseau de neurones classiques, les neurones prennent en entrée toutes les sorties des couches précédentes qui y sont connectées, cependant dans le cas d'un CNN, la sortie d'un neurone convolutif est une image. La sortie à la position (i, j) d'un neurone convolutif $o_k^{(c)}$ prenant en entrée la sortie de la couche $(c-1)$ se définit comme :

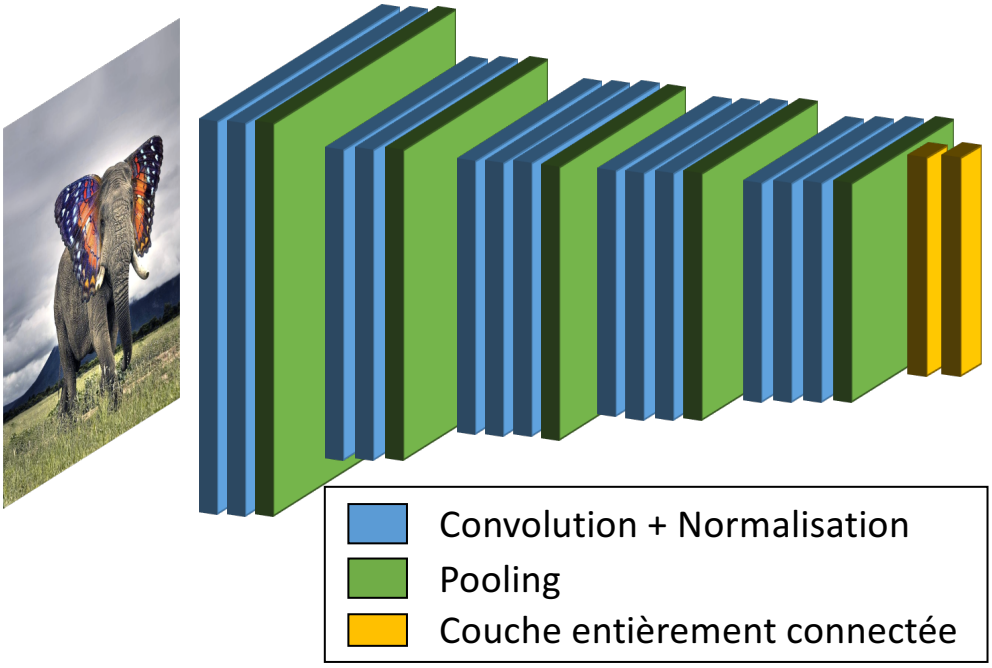


FIGURE 2.17 – Schéma d'un réseau de neurones convolutionnels.

$$o^{(c)}_{k_{(i,j)}} = b^{(c)}_k + \sum_{l=0}^{N^{(c-1)}} (w^{(c)}_k(j) * s^{(c-1)}_l) \tag{2.42}$$

Le biais $b^{(c)}_k$ du neurone convolutif est une constante qui est ajoutée à toutes les positions de l'image résultante.

Comme pour un neurone classique, une fonction d'activation notée f est appliquée sur la sortie du neurone :

$$s^{(c)}_k = f(o^{(c)}_k) \tag{2.43}$$

L'image $s^{(c)}_k$ en sortie du neurone est ce que l'on appelle une carte de caractéristiques ou *feature map*.

Il est aussi possible d'utiliser des convolutions dilatées [Yu et Koltun, 2015] ou aussi appelées convolutions atrous [Papandreou *et al.*, 2015]. L'idée de ce type d'opération est d'élargir le "champ de vision" de chaque neurone. Un paramètre r est utilisé pour déterminer le taux de dilatation, c'est-à-dire le pas avec lequel nous allons échantillonner le signal d'entrée. Lorsque $r = 1$ nous sommes dans le cas d'une convolution standard. Un exemple de convolution atrous où r correspond à la variable "rate" est présenté figure 2.18.

Ces couches de convolution vont permettre d'extraire ce que l'on appelle une carte de caractéristiques [Jarrett *et al.*, 2009] ou *feature map*. Afin de réduire la complexité calculatoire, ces réseaux utilisent le principe des poids partagés, cela signifie que plusieurs neurones vont utiliser le même noyau. Ce fonctionnement, en plus de réduire la complexité calculatoire (les noyaux ont généralement entre 9 et 121 paramètres), permet d'obtenir différentes invariances. En effet, par exemple cela nous permet de reconnaître le même

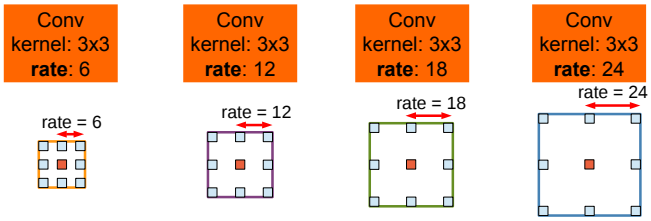
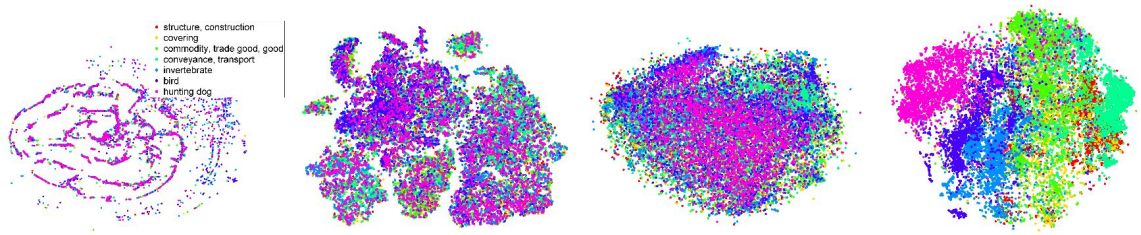


FIGURE 2.18 – Schéma représentant le fonctionnement d'une convolution atrous. Source : [Chen *et al.*, 2018].

motif quelque soit sa position dans l'image puisque le même noyau de convolution va être utilisé. En 2015, [Zhu *et al.*, 2015] ont montré de manière expérimentale qu'un nombre suffisamment élevé de couches de convolutions permet au réseau de devenir robuste aux orientations. Au cours de l'apprentissage les différentes couches de convolutions se spécialisent et forment un descripteur pertinent pour un problème donné.

La figure 2.19 montre la représentation de différents vecteurs caractéristiques en utilisant une projection 2D appelée t-SNE [Maaten et Hinton, 2008]. La t-SNE est une technique de réduction de dimensionnalité particulièrement adaptée à la visualisation d'ensembles de données de grande dimension. Les images (a) et (b) montrent la représentation de caractéristiques obtenues avec deux descripteurs, *Locality-constrained Linear Coding* [Wang *et al.*, 2010] (LLC) et GIST [Oliva et Torralba, 2001] respectivement, et les images (c) et (d) les *feature maps* extraites à la première couche d'un CNN et à la 6$^{\text{ème}}$ couche respectivement. On peut constater sur les images (a), (b) et (c) que les objets des différentes classes sont mélangés, la représentation d'un objet ne forme pas une zone homogène, les caractéristiques utilisées ne permettent donc pas de séparer les différentes classes. On voit que la première couche du réseau ne crée pas de descripteur assez pertinent pour séparer les classes. Cependant la sortie de la 6$^{\text{ème}}$ couche du CNN permet de regrouper les caractéristiques d'un même objet entre elles. Au cours de l'apprentissage les différentes couches de convolutions se spécialisent et forment un descripteur pertinent pour le problème donné.



(a) LLC (b) GIST (c) DeCAF$_1$ (d) DeCAF$_6$

FIGURE 2.19 – Représentation de la projection 2D t-SNE [Maaten et Hinton, 2008] de vecteurs caractéristiques. (a) est la représentation de caractéristiques de type LLC [Wang *et al.*, 2010], (b) de type GIST [Oliva et Torralba, 2001], (c) représente les *feature maps* extraites à la première couche d'un CNN et (d) à la 6$^{\text{ème}}$ couche. Source : [Donahue *et al.*, 2014].

2.2.3.1 L'étape de sous-échantillonnage

Ces types de réseaux sont dotés en plus d'une couche de sous-échantillonnage ou *pooling*. Cette couche permet de regrouper et de fusionner les valeurs des *feature maps*. La couche de sous-échantillonnage est une couche de transition, c'est-à-dire qu'elle ne possède pas de poids à mettre à jour. Cette couche applique une fenêtre glissante de taille $w \times h$ avec un pas de chevauchement donné sur toute la *feature map*. À chaque position, une opération de lissage attribue une valeur au contenu de la fenêtre glissante. Les deux méthodes les plus utilisées pour réaliser ce sous-échantillonnage sont soit la moyenne soit le maximum. L'utilisation de la moyenne permet de considérer toutes les valeurs de la *feature map* pour une fenêtre glissante donnée. Lors de la rétropropagation, l'erreur est répartie entre les différentes valeurs de la fenêtre glissante. L'utilisation du maximum ne conserve que la valeur la plus élevée de la fenêtre glissante donnée. Lors de la rétropropagation, l'erreur n'est logiquement propagée que sur la position du pixel avec la valeur la plus élevée. Les auteurs de [Scherer *et al.*, 2010] ont montré que l'utilisation du sous-échantillonnage par le maximum permet de converger plus rapidement lors de la phase d'apprentissage. De plus, utiliser le maximum permet d'introduire une robustesse aux translations.

Cette étape de sous-échantillonnage permet de faire aussi une réduction de la dimension. En effet, lorsque cette couche est utilisée le pas de chevauchement est généralement supérieur à un, cela veut dire que l'on va diminuer la taille de représentation des *feature maps*. Cette réduction permet un gain calculatoire non négligeable. Cependant, lorsque l'on applique un sous-échantillonnage, on perd une partie de l'information ce qui peut réduire les performances. Pour cette raison, le pas utilisé est petit (2 pixels), et l'utilisation abusive de cette couche est à éviter [Springenberg *et al.*, 2015].

Il existe également d'autres méthodes pour réaliser ce sous-échantillonnage. Par exemple le *Spatial Pyramid Pooling* (SPP) [Grauman et Darrell, 2005; Lazebnik *et al.*, 2006; He *et al.*, 2014] permet d'avoir en sortie un vecteur de taille fixe quelque soit la taille de l'entrée. Cette méthode est intéressante car si les couches de convolution peuvent avoir en entrée des images de tailles variables, le vecteur en entrée d'une couche *fully connected* (une couche connectée avec tous les neurones de la couche précédente, et tous les neurones de la couches suivante) doit être de taille fixe. Cette méthode permet donc d'avoir des images de taille variable en entrée du réseau pendant la phase d'apprentissage et de test.

Le principe de cette approche est d'utiliser un nombre de régions R fixe sur les images quelle que soit leur taille. La taille des régions R est donc proportionnelle à la taille de l'image d'entrée. Enfin, pour obtenir un vecteur de taille fixe, on applique un *pooling* sur chacune des régions.

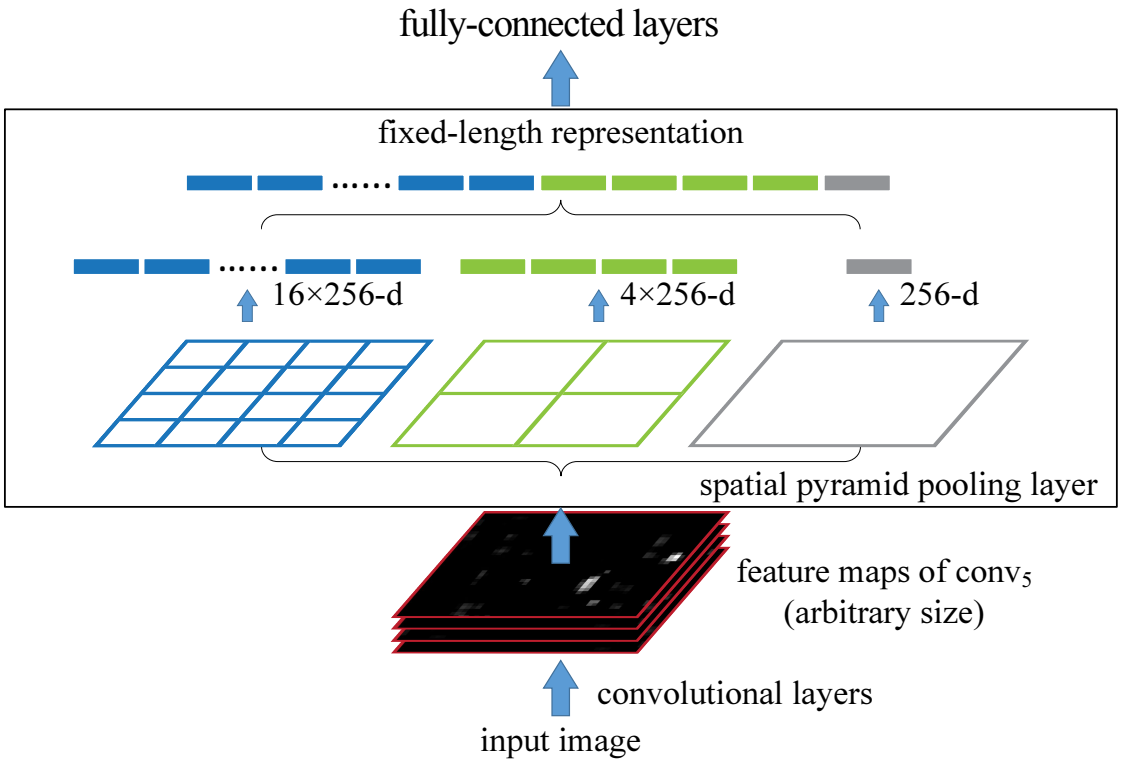


FIGURE 2.20 – Schéma représentant le fonction du SPP. En bas du schéma, nous avons les *features maps* en sortie de la dernière couche de convolution qui est composée de 256 noyaux. La partie dans le cadre représente le fonctionnement du SPP, où nous appliquons un pooling sur R régions des *feature maps*. Par exemple en gris, R = 1, une seule valeur va donc être extraite. La taille des région de R sont proportionnelles à la taille de la *feature maps*.

2.2.3.2 La normalisation

Une autre étape importante dans ces réseaux est la normalisation. En effet, les noyaux de convolution étant initialisés aléatoirement, il est possible que certaines valeurs des *feature maps* soient trop élevées comparées à celles des autres *feature maps*. Il est donc important de normaliser les différentes *feature maps* entre elles. Chaque *feature map* est normalisée en fonction des valeurs des autres *feature maps*.

Très fortement inspirés du fonctionnement de neurones biologiques [Lyu et Simoncelli, 2008], les auteurs de [Jarrett *et al.*, 2009] proposent de normaliser en soustrayant le résultat de la somme des *feature maps* avec un noyau gaussien $\mathbf{K} \in \mathbb{R}^{k_1 \times k_2}$ (k_1 la hauteur et k_2 la largeur de $\mathbf{K}$) à chaque *feature maps*. Cette normalisation est appelée normalisation par contraste local. Considérons la normalisation la couche c, la *feature map* après la normalisation $s_k^{(c)}$ du neurone $p_k^{(c)}$ est définie comme :

$$s_k^{(c)} = s_k^{(c)} - \sum_{l=0}^{\mathrm{N}^{(c)}} \mathbf{K} * s_k^{(c)} \tag{2.44}$$

En se basant sur la normalisation définie ci-dessus, une autre normalisation appelée normalisation par luminosité a été proposée [Krizhevský *et al.*, 2012]. L'idée est de normaliser la valeur à la position (x, y) en tenant compte de la valeur des n *feature maps*

adjacentes à la position (x, y). $s_k^{(c)}(x, y)$ est la valeur de la *feature map* à la position (x, y) du neurone $p_i^{(c)}$ et n le nombre de *feature maps* à considérer. La normalisation par luminosité se définie comme :

$$s_k^{(c)}(x, y) = \frac{s_k^{(c)}(x, y)}{\left(k + \alpha \sum_{j=max(0,k-n/2)}^{min(\mathrm{N}^{(c)}, \nu-n/2)} (s_j^{(c)}(x, y))^2\right)^{\beta}} \tag{2.45}$$

avec ν, α et β, des hyperparamètres qui sont déterminés avec la base de validation. Le phénomène de saturation étant rare grâce à l'utilisation de la fonction d'activation ReLU. En effet la fonction d'activation ReLU a la propriété de ne pas nécessiter de normalisation sur ses entrées pour empêcher la saturation. Grâce à cela, il n'est pas nécessaire de tenir compte de tous les noyaux d'une même couche. De plus, ne tenir compte que de n voisins permet de réduire le coût de calcul.

Cette normalisation se base sur un concept qui nous vient de la neurobiologie appelé «inhibition latérale». Cela fait référence à la capacité d'un neurone avec une forte activation à soumettre ses voisins. Cela tend à créer un contraste dans cette zone, augmentant ainsi la perception sensorielle, ce qui est une bonne chose et c'est pourquoi nous voulons avoir la même chose dans nos réseaux.

En 2015, une nouvelle normalisation permettant de régler le problème de saturation des noyaux tout en étant robuste aux différentes distributions en entrée [Ioffe et Szegedy, 2015] a vu le jour. Cette normalisation se sert de la statistique de la *feature map* pour normaliser chaque caractéristique. Pour que cette normalisation il est impératif de réaliser un apprentissage stochastique, c'est-à-dire par paquets, puisque l'ensemble de paramètres statistiques est calculé sur chaque paquet. Cette approche par paquets lui a valu le nom de *batch normalization* (BN). Les paramètres statistiques utilisés par cette normalisation sont la moyenne et la variance. Ces deux paramètres statistiques vont être calculés sur chaque valeur des *feature maps*. Ces deux valeurs statistiques sont ensuite moyennées avec l'ensemble des moyennes et variances calculées sur tous les paquets passés durant la phase d'apprentissage. Notons $\mu_k^{(c)}(x, y)$ et $\sigma_k^{(c)}(x, y)$ la moyenne et la variance de la *feature map* $s_k^{(c)}$ à la position (x, y).

$$s_k^{(c)}(x, y) = \frac{s_k^{(c)}(x, y) - \mu_k^{(c)}(x, y)}{\sqrt{\sigma_k^{(c)}(x, y)}} \tag{2.46}$$

Pour normaliser les *feature maps* entre elles, les auteurs de [Ioffe et Szegedy, 2015] proposent d'utiliser deux hyperparamètres γ et β qui vont être mis à jour lors de la phase d'apprentissage. Ces deux hyperparamètres sont calculés sur chaque *feature map*. Le calcul de la BN en prenant en compte γ et β est comme suit :

$$s_k^{(c)}(x, y) = \frac{\gamma}{\sqrt{\sigma_k^{(c)}(x, y)}} . \sigma_k^{(c)}(x, y) + \left(\beta - \frac{\gamma . \mu_k^{(c)}(x, y)}{\sqrt{\sigma_k^{(c)}(x, y)}}\right) \tag{2.47}$$

Les auteurs de [Wu *et al.*, 2015] montrent que l'utilisation de la BN leur permet de gagner 1% sur leur performances sur la base de données ImageNet. Cette normalisation est présente dans de nombreuses architectures de réseaux récents [Kaiming *et al.*, 2016; Gu *et al.*, 2017; Chen *et al.*; Zhao *et al.*, 2017].

2.3 Application à la détection d'objets

La tâche de localisation (appelée aussi détection) est une tâche dont le but est de détecter automatiquement différents objets dans des images ou des vidéos. Le schéma traditionnel pour réaliser cette tâche est présenté figure 2.21.

Le problème de détection d'objets peut être défini comme un problème d'étiquetage basé sur des modèles d'objets. Étant donné une image contenant un ou plusieurs objets d'intérêt et un ensemble d'étiquettes correspondant à un ensemble de modèles connus du système, le système devrait attribuer les étiquettes correctes dans les régions de l'image. Le terme de détection a été utilisé pour désigner de nombreuses capacités visuelles différentes y compris l'identification, la catégorisation et la discrimination. Dans notre cas nous parlons de détection ou de localisation lorsque l'on cherche à connaître la classe et la position ou la région d'un ou plusieurs objets dans une image. Les régions démarquant les objets sont localisées par ce que l'on appelle des boîtes englobantes (ou *bounding boxes* en anglais).

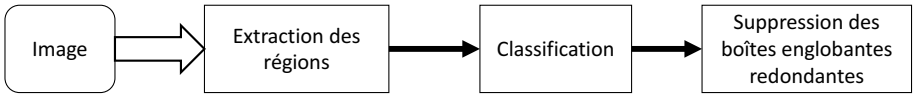


FIGURE 2.21 – Schéma classique d'une méthode de détection d'objets.

L'étape d'extraction des régions consiste à obtenir la localisation de l'objet qui peut apparaître n'importe où dans l'image. La taille et l'aspect de l'objet ne sont pas forcément connus, il faut donc utiliser une fenêtre glissante avec une série d'échelle sur l'ensemble de l'image. Cette stratégie exhaustive comprend l'ensemble des emplacements possibles de l'objet, mais il y a aussi un inconvénient à utiliser une telle stratégie : sa grande complexité en temps. Ensuite nous allons classer chacune des régions extraites pour déterminer si oui ou non l'objet que l'on cherche à détecter est dans une des régions. Pour cela, on utilise une approche d'apprentissage automatique en deux étapes :

Étape 1 : un réseau de neurones ou n'importe quelle autre technique peut être utilisée pour détecter la présence d'un objet dans une région.

Étape 2 : on supprime les boîtes englobantes redondantes. En effet, utiliser une fenêtre glissante à différentes tailles va créer de la redondance puisque l'on va parcourir entièrement toute l'image plusieurs fois. Pour cette étape, la méthode la plus utilisée est la *Non-Maximum Suppression* [Kitchen et Rosenfeld, 1982] (NMS). Si deux boîtes englobantes ont un ratio de chevauchement supérieur à un seuil, on supprime celle qui a le score le moins élevé.

La fenêtre glissante de détection d'objets a reçu une attention remarquable car elle est considérée comme une méthode très basique de détection d'objets dans une image ou une vidéo. La fenêtre glissante fonctionne essentiellement en cherchant à travers toute l'image ou la scène afin de trouver l'objet qui présente un intérêt. Ainsi, la complexité calculatoire est énorme, c'est pourquoi, aujourd'hui cette approche est très peu utilisée car on lui préfére des approches reposant sur le même principe mais plus rapides.

Après le succès en 2012 des CNN à la compétition ILSVRC, plusieurs chercheurs se sont tournés vers ce type de solution pour le problème de détection d'objets. En 2014, les auteurs de [Girshick *et al.*, 2014] proposent le *R-CNN* (Régions avec caractéristiques CNN). Le R-CNN a permis d'améliorer les résultats en matière de détection de manière conséquente. En effet, cette méthode montre que sur la base de données PASCAL VOC 2010, elle surpasse les méthodes existantes en apportant un gain de plus de 10% sur la précision moyenne de détection. La méthode utilisée pour arriver à ces résultats est la suivante : tout d'abord les auteurs commencent par extraire des régions de l'image susceptibles de contenir un objet, ensuite ils extraient un vecteur de caractéristiques de chaque région en utilisant le réseau de [Krizhevský *et al.*, 2012], et enfin ils utilisent un ensemble de SVM linéaires.

C'est à partir de l'article décrit précédemment que de nombreux travaux de recherche se sont basés sur des réseaux de neurones convolutionnels pour effectuer des tâches de détection [Jang *et al.*, 2015; Guo *et al.*, 2016; Portaz *et al.*, 2018]. En effet, on peut noter l'apparition du *Fast R-CNN* [Girshick, 2015] tout juste un an après celle du R-CNN. La différence majeure avec cette nouvelle "version", c'est que l'extraction des vecteurs caractéristiques et la classification des régions se font dans une seule et même structure. Cette nouvelle méthode prend en entrée une image et une liste de régions d'intérêt, et possède deux sorties, une pour nous donner la classe d'appartenance d'une région, et la deuxième sortie correspond à une regression qui est appliquée sur les coordonnées de la région d'intérêt. Cette deuxième sortie permet de mieux cadrer l'objet présent dans une région d'intérêt. Cette approche utilise deux sorties, et donc une fonction de perte multi-tâches. En effet, afin d'obtenir de meilleures performances, le réseau doit minimiser la fonction objectif sur la tâche de classification, mais il doit aussi améliorer le placement des régions d'intérêt en minimisant la distance entre la région générée et celle de la vérité terrain. Cette fonction de perte est de la forme suivante :

$$\mathrm{L}(\hat{t}, t, \hat{v}^u, v) = \mathrm{L}_{cls}(\hat{t}, t) + \lambda[t \geq 1]\mathrm{L}_{loc}(\hat{v}, v) \tag{2.48}$$

avec $\mathrm{L}(\hat{t}, t, \hat{v}^u, v)$ la fonction de perte globale, $\mathrm{L}_{cls}(\hat{t}, t) = -log\mathrm{P}(t|\hat{t})$, la fonction de perte pour la classification avec $\hat{t}$ la classe prédite et t la classe attendue. La deuxième partie de la fonction de perte globale correspond à la fonction de régression L_{loc} pour améliorer la localisation. L_{loc} est définie à partir des coordonnées de la boîte englobante de la vérité $\hat{v}$ et des coordonnées prédites v. Le terme $[t \geq 1]$ prend la valeur 1 si t est différent de la classe "autre", sinon il prend la valeur 0 et la fonction de régression n'est donc pas prise en compte dans la fonction de perte globale. La fonction de régression

utilisée pour la localisation est la suivante :

$$\mathrm{L}_{loc}(\hat{v}, v) = smooth_{\mathrm{L1}}(\hat{v} - v) \tag{2.49}$$

avec

$$smooth_{\mathrm{L1}}(x) = \begin{cases} 0.5x^2 & \text{Si } |x| < 1 \\ |x| - 0.5 & \text{sinon} \end{cases} \tag{2.50}$$

De cet article a découlé une nouvelle méthode appelée *Faster R-CNN* [Ren *et al.*, 2015]. La nouveauté de cette méthode est quelle intègre la dernière étape qui n'était pas encore intégrée au réseau, c'est-à-dire la génération des régions d'intérêt. Pour générer ces régions, les auteurs utilisent un réseau entièrement constitué de couches convolutionnelles qui sont partagées avec un réseau effectuant la classification et la régression comme dans le cas du *Fast R-CNN*. Une fenêtre glissante est appliquée sur la *feature map* de la dernière couche de convolution partagée et sur cette fenêtre k régions sont extraites. Pour chaque fenêtre le réseau aura donc $4k$ sorties correspondant aux coordonnées des boîtes englobantes extraites sur lesquelles la fonction de régression va être appliquée, et $2k$ sorties qui vont servir à minimiser la fonction de perte de la classification.

2.4 Application à la segmentation sémantique

La segmentation est une opération de traitement d'images dont le but est de rassembler des pixels entre eux en suivant des critères pré-définis. Les pixels sont regroupés en régions qui constituent un pavage ou une partition de l'image. Le but est de rassembler les pixels en régions homogènes, par exemple pour séparer un objet du fond. Dans notre cas il s'agit de segmentation sémantique, cela veut dire que nous connaissons à l'avance le nombre de classes présentes dans l'image.

Il existe de nombreuses méthodes de segmentation qui peuvent être divisées en deux familles :

— La segmentation basée sur les contours,

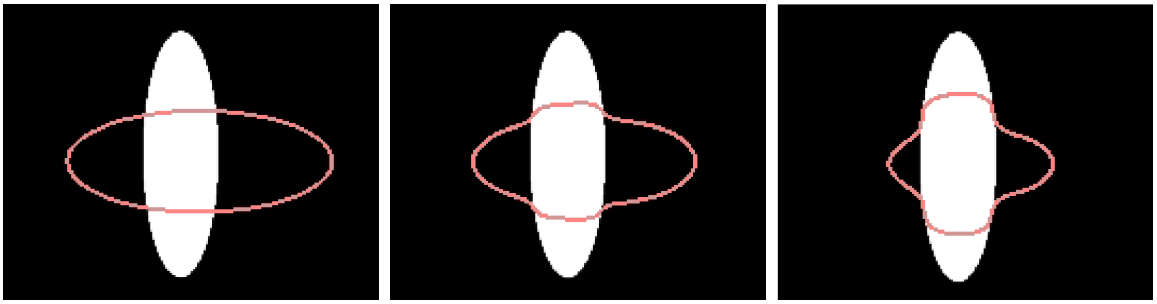
— La segmentation basée sur les régions.

Il existe aussi des méthodes de segmentation hybrides qui utilisent les deux familles citées précédemment.

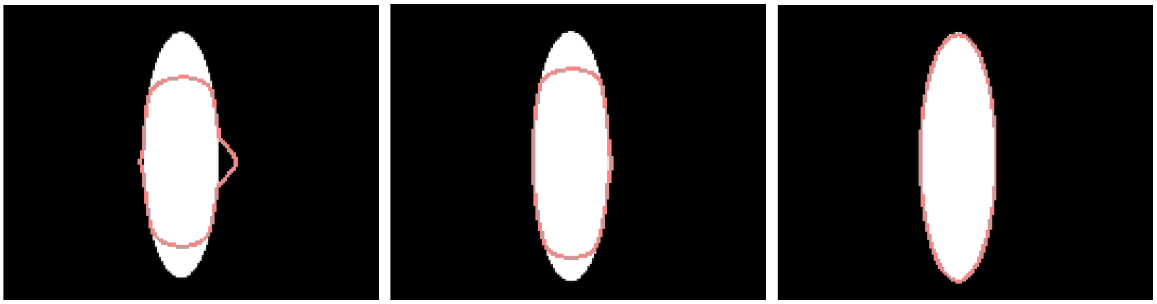
La segmentation basée sur les contours s'intéresse aux contours des objets dans l'image. La majorité de ces algorithmes appartenant à cette famille de segmentation sont locaux, c'est-à-dire qu'ils travaillent au niveau des pixels. Les principaux problèmes liés à cette approche sont qu'il faut dans un premier temps caractériser les frontières entre les objets, mais aussi que les contours sont souvent incomplets et il faut donc être capable de les fermer. De nombreux opérateurs existent afin de détecter les contours, comme par exemple celui de [Roberts, 1963] ou bien celui de [Prewitt, 1970]. Plus récemment [Martin *et al.*, 2004] ont défini des opérateurs de gradient pour la luminosité, la couleur et la texture, et les utilisent comme entrée d'une régression logistique afin de prédire les contours. Ces

méthodes ont maintenant été améliorées [Arbelaez *et al.*, 2011] notamment en utilisant des informations multi-résolutions [Ren, 2008].

Une autre approche utilisée en segmentation basée sur les contours est appelée l'approche par contours actifs [Kass *et al.*, 1988; Terzopoulos *et al.*, 1987]. Pour former ce contour actif, on utilise une série de points sur une courbe C que l'on va déformer au cours des itérations (voir figure 2.22). Pour ce faire, on va chercher à minimiser une énergie qui se compose : d'une énergie d'attache aux données (qui permet d'attirer les contours vers les caractéristiques de l'image en utilisant le gradient de l'image par exemple), d'une énergie de régularisation (qui permet de contraindre la forme et l'évolution du contour), et d'une énergie permettant d'ajouter des contraintes sur la solution.



(a) 1ère iteration : contour initial (b) Iteration 31 (c) Iteration 81



(d) Iteration 151 (e) Iteration 191 (f) Iteration 353 : contour final

FIGURE 2.22 – Déformation d'un contour actif (ellipse rose) vers un objet (ellipse blanche) au cours des itérations. Source : [Gastaud, 2005].

Les approches de segmentation basées régions se basent sur différents critères, comme des critères spatiaux ou bien temporels pour trouver des régions homogènes. La méthode de segmentation basée sur les régions la plus connue est une segmentation par ligne de partage des eaux [Beucher, 1979]. Cette méthode se base sur la morphologie mathématique qui considère une image en niveaux de gris comme un relief topologique sur lequel on simule une inondation. L'idée de base consistait à placer une source d'eau dans chaque minima du relief, puis simuler une montée des eaux et enfin à construire des barrières lorsque différentes sources d'eau se rencontrent. Les bassins versants obtenus cor-

respondent aux régions de la partition. De nombreuses améliorations ont été apportées à cet algorithme [Barnes *et al.*, 2014], notamment sur le placement des sources d'eau. Ce type de méthode est encore très largement utilisé, notamment en médecine et en télédétection [Anand, 2017; Goldbergs *et al.*, 2018].

Une autre méthode de segmentation par région est appelée segmentation par ensemble de niveaux [Osher et Sethian, 1988] ou *level-set*. L'idée de cette méthode est de considérer une grille sur laquelle on connaît la distance de chaque point par rapport au contour actif. L'idée n'est donc plus de déplacer le contour actif mais de modifier la valeur des points de la grille. Lorsque les points ont une distance nulle, cela signifie qu'ils sont sur le contour solution. Généralement pour calculer la distance des points par rapport à la courbe solution, on utilise la distance euclidienne avec un signe positif pour les points à l'extérieur de la courbe et un signe négatif pour les points à l'intérieur de la courbe.

La segmentation par classification a énormément évolué au fil du temps, notamment avec l'apparition de l'apprentissage profond. À l'origine ce type de méthode était généralement utilisé en plus d'une autre méthode. Par exemple [Song et Civco, 2004] se servent dans un premier temps d'un SVM pour faire une segmentation grossière des images en deux classes, la classe route et la classe autre. Puis ils utilisent une approche de croissance de région à partir du résultat obtenu par le SVM. Cette approche consiste à faire grossir une région tant qu'elle continue de remplir un critère de similarité ou bien en optimisant une fonction objectif. Cependant il existe aussi des méthodes utilisant uniquement une approche d'apprentissage automatique comme [Bertelli *et al.*, 2011] qui utilise le descripteur HOG avec un SVM.

L'utilisation du *deep learning* pour la segmentation a commencé dès 2009 avec [Grangier *et al.*, 2009]. De plus en plus d'études se sont concentrées sur des réseaux convolutifs de plus en plus profonds pour réaliser des tâches de segmentation et les résultats étaient très prometteurs. En 2012 [Socher *et al.*, 2012] proposent d'utiliser dans un premier temps une seule couche de convolution et un *pooling*. Cette première étape leur sert d'extracteur de caractéristiques. Sur cet extracteur, ils empilent des réseaux neuronaux récursifs (RNN) qui sont utilisés sur des blocs locaux du résultat de l'extracteur.

En 2013 [Farabet *et al.*, 2013] réalisent l'apprentissage de CNNs avec trois couches de convolution et une couche entièrement connectée. Cependant, prédire la classe de chaque pixel indépendamment de son voisinage mène à un résultat bruité. Afin de débruiter les résultats obtenus et d'améliorer les performances il faut forcer les régions de même intensité de couleur à avoir une seule classe. Pour cela les auteurs ont testé deux approches : l'approche par superpixels [Felzenszwalb et Huttenlocher, 2004] et l'utilisation d'un Champ Aléatoire Conditionnel [Lafferty *et al.*, 2001] (*Conditional Random Field* en anglais ou CRF). Le but de l'approche par superpixels est de produire une sursegmentation et ensuite d'agréger les prédictions des pixels dans chaque superpixel en calculant par exemple la répartition moyenne des classes dans le superpixel. Les CRF eux sont des modèles statistiques permettant de prendre en compte l'interaction des variables voisines. Pour le cas de la segmentation, ils permettent de prendre en compte les

voisins et ainsi d'éviter d'avoir un pixel "isolé", c'est-à-dire un pixel dont le voisinage n'a pas la même classe.

Ces méthodes ont énormément évolué aujourd'hui, notamment avec [Chen *et al.*, 2018] qui proposent d'utiliser des convolutions atrous, permettant d'augmenter le champ de vision des filtres sans augmenter le nombre de paramètres. Les auteurs proposent aussi dans cet article un nouveau *pooling*, appelé *Atrous Spatial Pyramid Pooling* et inspiré du *Spatial Pyramid Pooling* vu à la section précédente. Ce *pooling* permet de segmenter les objets à plusieurs résolutions en utilisant des convolutions atrous avec des "rate" différents. Le résultats de ce *pooling* est donné à un CRF afin de raffiner les résultats. On peut aussi noter que les réseaux qu'ils ont utilisés sont nettement plus profonds qu'auparavant puisqu'il s'agit des réseaux VGG-16 [Simonyan et Zisserman, 2015] et ResNet-101 [Kaiming *et al.*, 2016] qui sont respectivement composés de 16 et 101 couches de convolution. Les résultats de cet article obtenus sur la base de données cityscapes [Cordts *et al.*, 2016] sont présentés figure 2.23. Cette figure présente l'image testée, la vérité terrain, le résultat à la sortie du réseau avant d'appliquer le CRF et après le CRF. On peut constater que les résultats après le CRF sont plus lissés sur les contours et que le nombre de pixels isolés a grandement diminué.

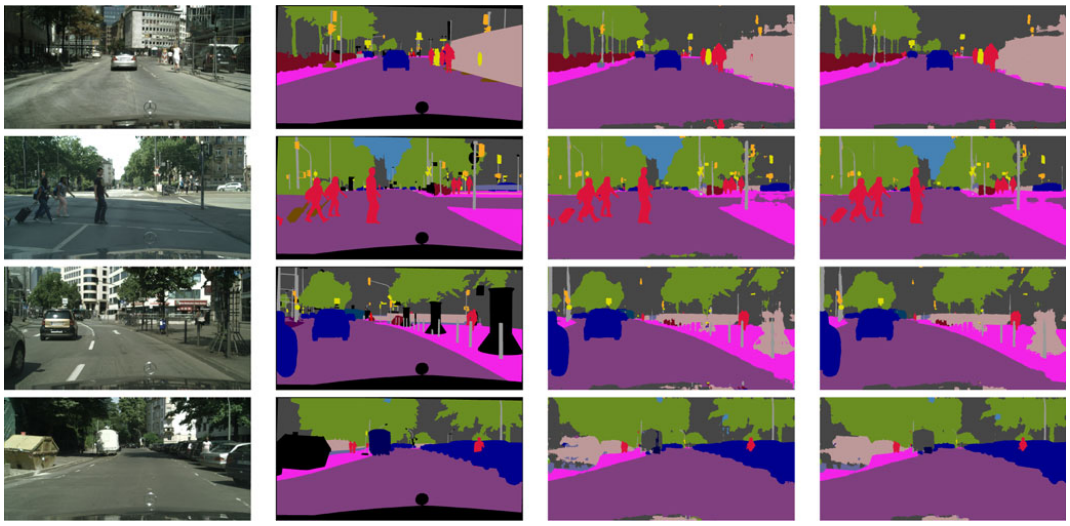


FIGURE 2.23 – Résultats sur la base cityscapes [Cordts *et al.*, 2016]. Les colonnes de gauche à droite représentent : l'image testée, la vérité terrain, le résultat avant le CRF et le résultat après le CRF. Source [Chen *et al.*, 2018].

2.5 Le problème de la fusion de données et de la gestion des données manquantes

2.5.1 La fusion de données

La technologie concernant les capteurs ayant grandement évolué et leur prix étant de plus en plus bas, de plus en plus d'études se concentrent sur des méthodes de fusion afin

d'améliorer les performances de leurs classifieurs [Kanaev *et al.*, 2011]. En effet, combiner les résultats de plusieurs capteurs peut fournir des informations plus précises que l'utilisation d'un seul capteur [Chong *et al.*, 2000; Waltz et Llinas, 1990; Schmitt et Zhu, 2016; Hedhli *et al.*, 2017] et permet une précision améliorée ou la même performance mais avec des capteurs plus petits ou moins chers.

Les techniques de fusion de données combinent les données de plusieurs capteurs, et des informations connexes à partir des bases de données associées afin d'atteindre les meilleurs résultats possibles et de rendre leur utilisation aussi simple qu'avec l'utilisation d'un seul capteur [Waltz, 1986; Hall, 1992]. Bien que le concept de fusion de données n'est pas nouveau, l'émergence de nouveaux capteurs, de techniques de traitement avancées et de matériel de traitement amélioré ont permis de rendre la fusion de données en temps réel possible [Llinas et Hall, 1994; Hall et Llinas, 1994; Hedhli *et al.*, 2017].

Les applications liées à la fusion de données sont très variées. À l'origine ces applications étaient réservées à l'armée du fait de leur accès à différentes données. D'un point de vue purement militaire, on peut noter les applications suivantes : reconnaissance automatique de cible, guidage pour les véhicules autonomes, télédétection, surveillance du champ de bataille et automatisation des systèmes de reconnaissance des menaces [Hall *et al.*, 1991]. Très rapidement d'autres applications non militaires sont apparues, comme la surveillance des processus de fabrication, la robotique [Abidi et Gonzalez, 1992] ou bien des applications médicales.

La fusion de plusieurs capteurs était initialement réalisée en utilisant un filtre de Kalman [Kalman, 1960; Gao et Harris, 2002] ou bien un filtre bayésien. Plusieurs travaux ont porté sur une approche d'apprentissage automatique en deux étapes, notamment dans le domaine de la télédétection [Dalponte *et al.*, 2012; Liu et Bo, 2015]. Lorsque l'on parle de fusion de données deux approches sont souvent opposées : l'approche ***Early Fusion*** [Lagrange *et al.*, 2015; Paisitkriangkrai *et al.*, 2015; Liu *et al.*, 2017] (EF) et l'approche ***Late Fusion*** [Audebert *et al.*, 2016; Wagner *et al.*, 2016; **?**] (LF). L'approche ***Early Fusion*** consiste à combiner toutes les informations et ensuite les donner à un classifieur. Pour l'approche ***Late Fusion***, on va dans un premier temps traiter chaque information indépendamment, par exemple en réalisant un apprentissage sur chaque information séparémment, et ensuite on va combiner les scores ou les résultats obtenus sur chaque information. De nombreux travaux proposent de comparer ces deux approches [Snoek *et al.*, 2005; Wagner *et al.*, 2016; Petscharnig *et al.*, 2018; Audebert *et al.*, 2018] et montrent qu'il n'y a pas une approche plus efficace que l'autre. Par exemple les auteurs de [Audebert *et al.*, 2018] montrent que l'approche ***Early Fusion*** permet d'obtenir les caractéristiques les plus efficaces, mais que l'approche ***Late Fusion*** est capable de correctement classer des pixels où les autres méthodes se trompent.

Plus récemment, plusieurs travaux se sont penchés sur le problème de fusion de données avec l'utilisation du *deep learning* [Ngiam *et al.*, 2011]. Par exemple [Wagner *et al.*, 2016] proposent une nouvelle architecture de CNN pour fusionner des images RGB et l'image de profondeur en niveaux de gris. Le schéma de cet article est présenté figure 2.24.

Le principe de cette architecture est de séparer le traitement des données de type différent dans des branches différentes du réseau et de fusionner les branches avant d'effectuer la classification. Dans le schéma 2.24, la branche représentée dans le cadre bleu traite les images avec les canaux bleu, rouge et vert, et la branche dans le cadre rouge traite l'image de profondeur.

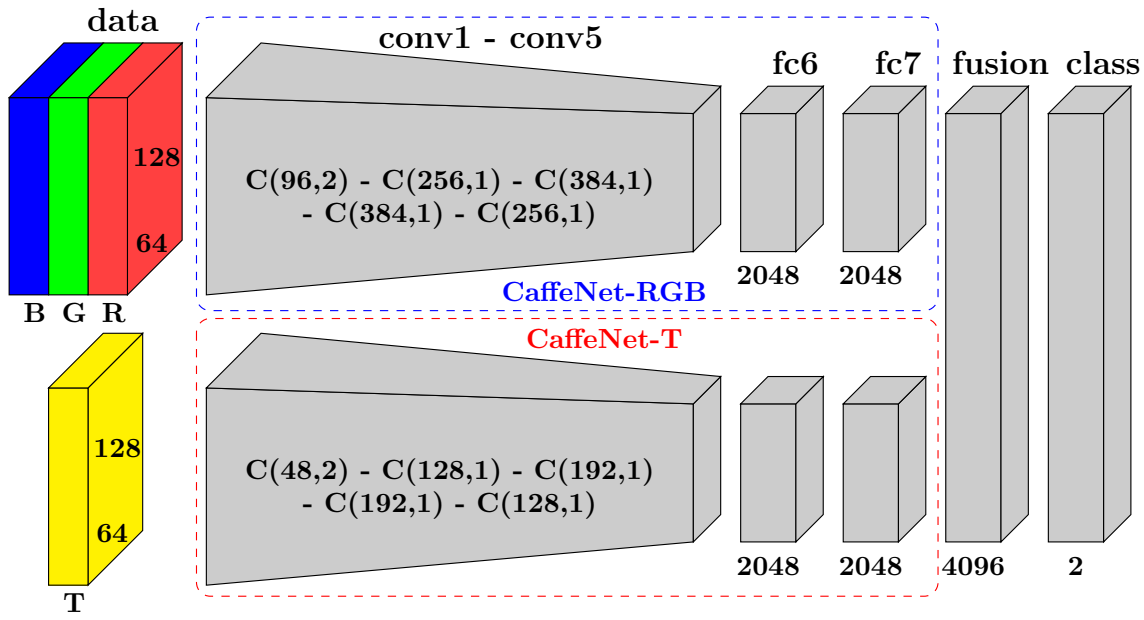


FIGURE 2.24 – Schéma de l'architecture de [Wagner *et al.*, 2016] permettant de faire de la fusion de données multi-sources dans un CNN. Dans le cadre bleu, les images optiques avec les canaux bleu, vert et rouge sont traitées, et dans le cadre rouge le réseau traite les images de profondeur.

Nous traiterons dans le chapitre 4 de la fusion et nous proposerons une modification de l'approche de [Wagner *et al.*, 2016].

2.5.2 Les données incomplètes

La plupart des travaux sur la fusion de données supposent que toutes les sources (images optiques, MNS, Pir...) sont disponibles pour chaque point de données d'apprentissage Di *et al.* [2010]; Lai *et al.* [2011]. Cette hypothèse est très limitative d'un point de vue pratique car il est très rare d'avoir accès à toutes les modalités sur un échantillon. Ce problème peut provenir d'une erreur lors de l'étalonnage qui peut rendre une modalité incompatible avec les autres. Lorsque nous manquons une modalité et que nous souhaitons effectuer une tâche en utilisant des données multimodales, nous avons plusieurs solutions : (i) supprimer la modalité incomplète, (ii) supprimer les échantillons qui n'ont pas toutes les modalités, (iii) ou générer des modalités manquantes sur les échantillons incomplets Tran *et al.* [2017]. Malheureusement, les deux premières solutions supprimeront beaucoup d'informations et donc des performances plus faibles.

La tendance générale tend vers la génération des données manquantes. Pour ce faire,

il est courant d'utiliser des modèles à variables latentes, comme l'analyse factorielle et les versions «plus profondes» telles que les auto-encodeurs [Kingma et Welling, 2013; Rezende *et al.*, 2014] (AE) et les *Generative Adversarial Networks* [Goodfellow *et al.*, 2014; Gauthier, 2014] (GAN).

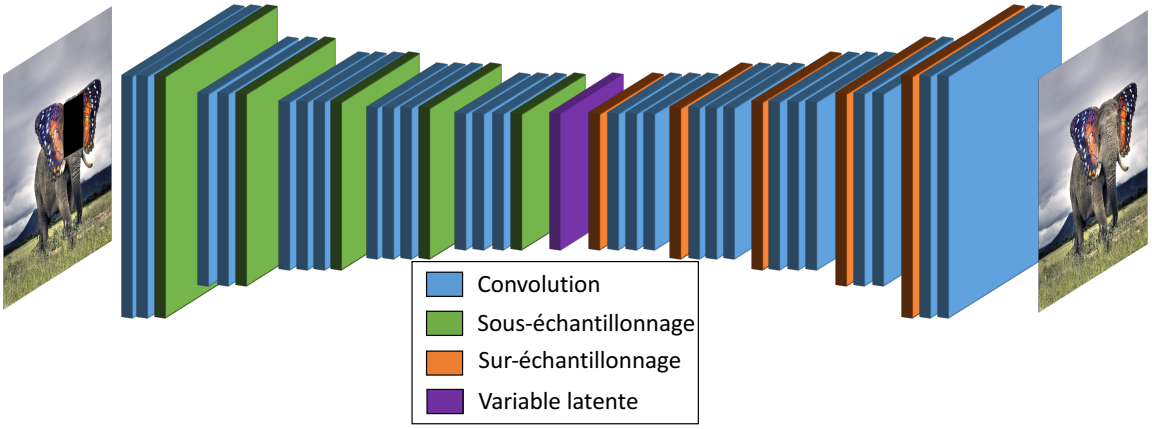


FIGURE 2.25 – Exemple d'auto-encodeur permettant de reconstruire une partie cachée dans une image.

Les auto-encodeurs sont composés de deux parties, une partie encodeur qui a pour but d'encoder les données fournies à l'auto-encodeur. L'objectif cette partie est de prédire la distribution des variables latentes à partir des données d'entrée. La deuxième partie est appelée le décodeur. Le rôle de cette partie est de faire correspondre les variables latentes aux variables d'entrée, c'est-à-dire les données que nous avons dans notre base d'apprentissage. L'apprentissage de ce type de réseau s'effectue par le biais de la rétropropagation en minimisant l'erreur quadratique présentée équation 2.26 de la même façon que pour un réseau de neurones classique.

La figure 2.25 présente un cas d'application des auto-encodeurs. Nous avons en entrée une image dont une partie est cachée par un rectangle noir et le but de l'auto-encodeur est de reconstruire la partie cachée. De nombreux travaux ont été menés sur le cas de données bruitées ou partiellement cachées [Xie *et al.*, 2012; Pathak *et al.*, 2016; Cho, 2013].

Le fonctionnement d'un GAN [Goodfellow *et al.*, 2014; Gauthier, 2014] est présenté figure 2.26. Comme on peut le voir sur la figure, un GAN est composé de deux parties, le générateur (en bleu) et le discriminateur (en orange). Le rôle du générateur est de générer de nouvelles données à partir d'un bruit aléatoire tout en essayant de tromper le discriminateur. Quant au discriminateur, son rôle est de déterminer si la donnée en entrée est une "vraie" image ou bien si elle a été générée. Comme pour les auto-encodeurs, l'apprentissage est réalisé en utilisant la rétropropagation. L'objectif ici serait de générer les sources manquantes sur certaines données de la base de données.

L'idée de ce type d'architecture n'est pas de réussir à reproduire les images de la base d'apprentissage, mais de générer de nouvelles images dont la distribution est proche des images de la base d'apprentissage.

Ces solutions permettent de faire fonctionner les alorithmes qui nécessitent l'ensemble

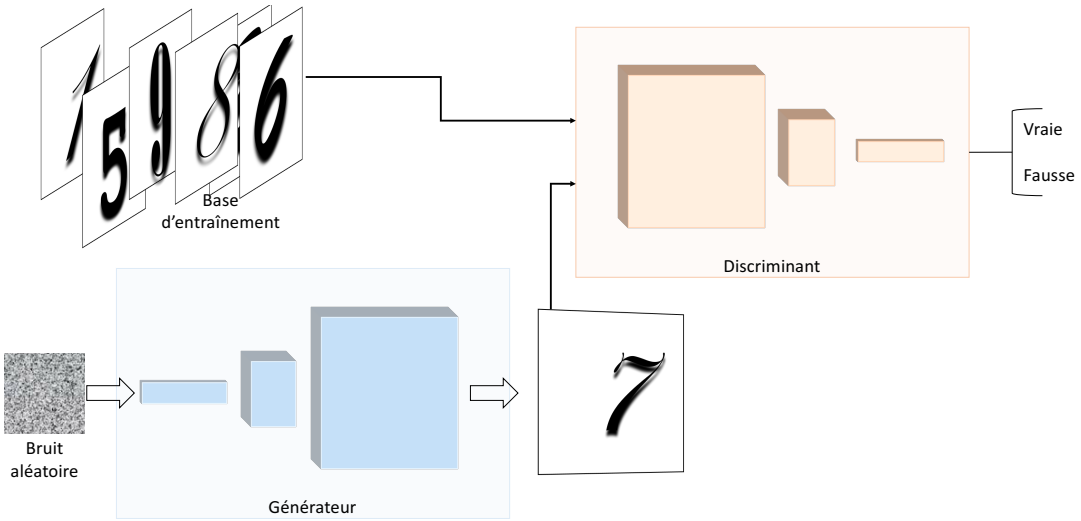


FIGURE 2.26 – Illustration du fonctionnement d'un GAN.

des modalités pour marcher, cependant elles ne sont pas complètement satisfaisantes car on génère une donnée synthétisée. Dans le chapitre 5 nous présentons une alternative à ces solutions.

Chapitre 3

Comparaison entre des méthodes d'apprentissage automatique classiques et du *deep learning*

Sommaire

3.1 Localisation des objets et fusion des résultats

La première étape de cette thèse a été de comparer les approches d'apprentissage automatique classiques, c'est-à-dire avec une première étape d'extraction des caractéristiques, puis l'utilisation d'un classifieur (voir chapitre 2) aux approches *deep learning* sur la tâche de détection d'objets urbains. Nous avons donc comparé des méthodes qui dans un premier temps utilisent un descripteur permettant d'extraire un vecteur de caractéristiques et ensuite utilisent un classifieur à une approche qui réalise l'extraction de caractéristiques et la classification en même temps. Nous présentons dans ce chapitre la méthodologie que nous avons utilisée afin de réaliser notre étude préliminaire sur les réseaux de neurones convolutionnels.

Nous proposons d'évaluer la capacité des approches CNN pour la tâche de détection d'objets dans les données aériennes multi-sources. Nous comparons les performances de CNN avec des méthodes d'apprentissage automatique en deux étapes qui exploitent des descripteurs conçus à la main. Nous allons comparer deux architectures CNN bien connues, AlexNet [Krizhevský *et al.*, 2012] et GoogLeNet [Szegedy *et al.*, 2015] et deux méthodes d'apprentissage automatique basées sur le descripteur d'image HOG en cascade (Histogram of Oriented Gradient) [Qiang *et al.*, 2006] avec deux classifieurs (SVM, Random Forests). Comme exemple d'application, nous considérerons la localisation et la détection d'arbres urbains dans des données aériennes multi-sources composées de données multispectrales (avec les canaux rouge, vert et proche infrarouge) et de modèles numériques de surface (MNS) des zones urbaines.

Un aperçu général de notre méthode est présenté dans la figure 3.1. Nous entraînons un classifieur pour discriminer la classe "arbre" de la classe "autre". La base d'apprentissage est composée d'imagettes ayant toutes la même taille (Figure 3.1 a). Dans ce chapitre, nous avons traité l'aspect multi-sources en utilisant une approche simple. Cette approche consiste à prendre chaque composante RV, Pir et MNS comme une composante d'une image. Pour la phase de test, une fenêtre glissante multi-résolutions est appliquée sur l'image de test. À chaque position, le contenu de la fenêtre glissante est ensuite donné au modèle appris afin d'obtenir un score d'appartenance à la classe "arbre" et "autre" (Figure 3.1 b).

Puisque la fenêtre glissante est appliquée à différentes résolutions, plusieurs prédictions vont être présentes sur une même zone. Nous fusionnons successivement toutes les boîtes englobantes classées comme "arbre" [Sermanet *et al.*, 2014] pour obtenir la localisation finale des arbres dans les images (Figure 3.1 c). Dans nos expériences, nous avons utilisé deux mécanismes différents pour réaliser les fusions : i) une fusion par aire [Sermanet *et al.*, 2014] et ii) une fusion par chevauchement [Broggi *et al.*, 2007].

Le reste de ce chapitre est organisé comme suit : la section 3.1.1 est consacrée à la présentation des algorithmes utilisés pour réaliser la détection. Dans la section 3.1.2 nous présentons les différents algorithmes que nous mettons en compétition. Dans la section 3.2.1 nous détaillons le protocole expérimental utilisé et finalement nous concluons

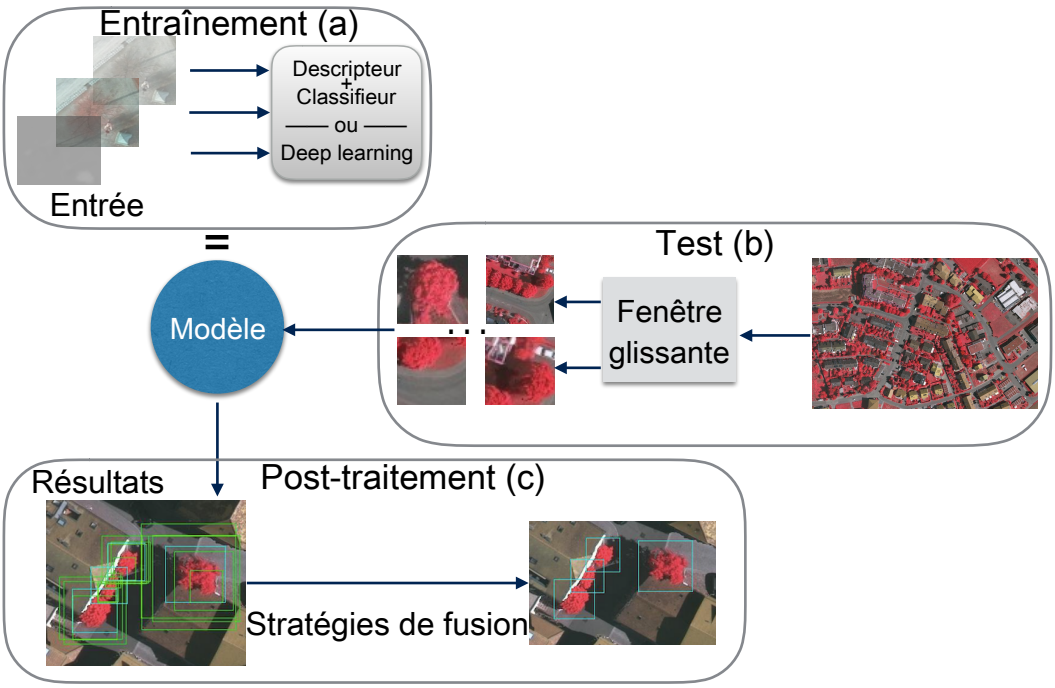


FIGURE 3.1 – Schéma de la méthode proposée. La partie (a) correspond à la phase d'apprentissage, c'est ici que l'on donne nos imagettes au classifieur afin qu'il apprenne à discriminer la classe "arbre" de la classe "autre". Lorsqu'il s'agit de la méthode par apprentissage automatique en deux étapes, on commence par extraire des vecteurs de caractéristiques à partir des imagettes et on donne ces vecteurs de caractéristiques à un classifieur. Lorsqu'il s'agit de l'approche *deep learning*, alors nous donnons directement les imagettes à notre CNN. La partie (b) représente la phase de test. C'est pendant la phase de test que l'on applique la fenêtre glissante multi-résolutions sur nos images de test afin de déterminer la localisation des arbres dans les images. Enfin la partie (c) correspond à l'application d'un post-traitement sur les résultats obtenus dans la partie (b). C'est ici qu'une stratégie de fusion des boîtes englobantes est appliquée afin de supprimer les boîtes redondantes et de réduire le nombre de faux positifs.

en section 3.3.

3.1.1 Fenêtre glissante et stratégies de fusion

Pour localiser les arbres sur les images testées, nous avons utilisé une fenêtre glissante multi-résolutions [Garcia et Delakis, 2004]. Afin de s'assurer de ne pas rater d'arbres (ou le moins possible), et d'avoir au moins une boîte englobante parfaitement centrée sur l'arbre, nous avons choisi de déplacer la fenêtre glissante avec un pas très faible (seulement 2 pixels). De plus, puisque le diamètre de la couronne des arbres peut grandement varier d'un arbre à un autre, cette méthode nous permet de détecter tous les arbres indépendamment du diamètre de leurs couronnes.

Comme les imagettes extraites par la fenêtre glissante doivent être de la même taille que les imagettes utilisées pendant la phase d'apprentissage, nous avons utilisé une fenêtre glissante de taille fixe et nous avons fait varier la résolution de l'image testée. Nous

redimensionnons chaque image de 30% jusqu'à 300% de leur taille d'origine avec un pas de 10%, nous avons donc 28 images sur lesquelles nous appliquons la fenêtre glissante. Les imagettes extraites par la fenêtre glissante sont données au modèle qui va nous fournir le score d'appartenance à classe "arbre". Ainsi, le modèle nous permet de déterminer les zones de l'image qui sont susceptibles de contenir un arbre, c'est-à-dire les zones qui ont un score d'appartenance à la classe "arbre" élevé. Ces zones susceptibles de contenir un arbre sont marquées par ce que l'on appelle des boîtes englobantes. Les boîtes englobantes sont placées en utilisant les coordonnées de la fenêtre glissante dont le score d'appartenance à la classe "arbre" est élevé. Afin de déterminer si on doit garder ou non une boîte englobante, nous avons utilisé un seuil que nous avons fixé en utilisant une base de validation. Pour régler ce seuil, nous avons choisi de prendre le seuil qui maximise la F-mesure sur notre base de validation. La figure 3.2 illustre le fonctionnement de la fenêtre glissante multi-résolutions.

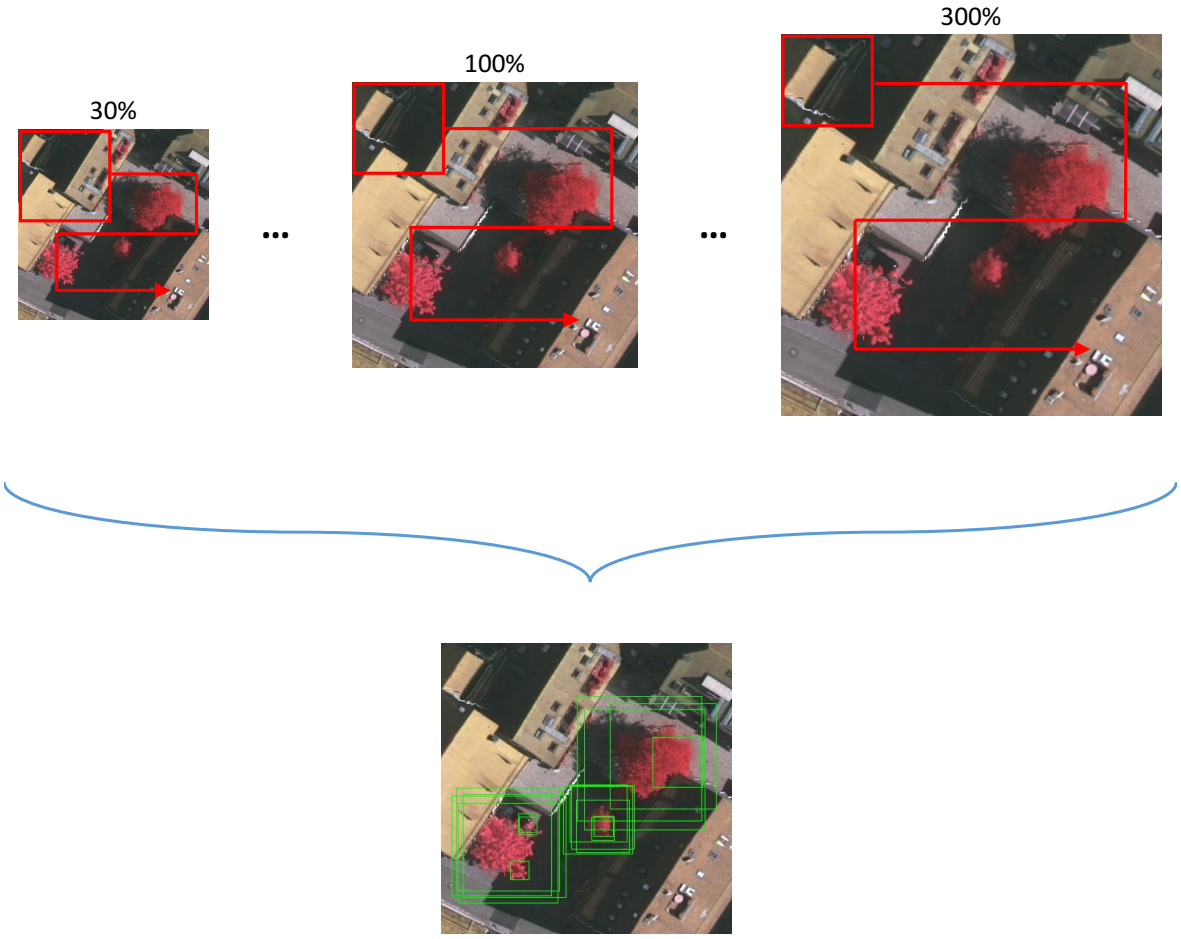


FIGURE 3.2 – Illustration de la fenêtre glissante sur une image. Sur la partie supérieure du schéma, nous faisons varier la résolution de l'image de 30% jusqu'à 300%. Nous appliquons ensuite une fenêtre glissante de taille fixe sur toutes les images. Dans la partie inférieure du schéma, nous montrons les boîtes englobantes qui ont été retenues après le passage de la fenêtre glissante multi-résolutions. Les boîtes englobantes sont représentées en vert sur l'image, il s'agit des zones de l'image qui ont un score d'appartenance à la classe "arbre" supérieur au seuil qui maximise la F-mesure. Nous avons fixé ce seuil en utilisant une base de validation

Comme on peut le constater sur la figure 3.2, l'application de notre fenêtre glissante multi-résolutions va créer une accumulation de boîtes englobantes sur une même zone. Pour surmonter ce problème, il faut fusionner l'ensemble des boîtes englobantes qui sont classées comme contenant un arbre. L'idée de cette fusion est bien sûr de supprimer les boîtes englobantes accumulées sur une même zone, mais on ne veut pas supprimer des boîtes qui correspondent à deux arbres qui sont proches. Nous avons utilisé deux stratégies différentes pour réaliser la fusion : i) la fusion par aire [Sermanet *et al.*, 2014] et ii) la fusion par chevauchement [Broggi *et al.*, 2007].

3.1.1.1 Fusion par aire

La fusion par aire a été employée dans [Sermanet *et al.*, 2014]. Cette fusion va comparer toutes les boîtes englobantes par paire. Pour chaque paire, nous calculons si l'indice de Jaccard, c'est-à-dire si l'aire de l'intersection des deux boîtes englobantes divisée par l'aire de l'union des deux boîtes est supérieure à 0,5 (voir équation 3.1). Si c'est le cas, la boîte englobante ayant le score le plus faible de contenir un arbre est supprimée.

$$\frac{Aire(B_1 \cap B_2)}{Aire(B_1 \cup B_2)} > 0,5 \tag{3.1}$$

avec B_1 et B_2 les deux boîtes englobantes.

Avec ce type de fusion, si B_1 est comprise dans B_2 mais que son aire est au moins deux fois plus petite, alors la condition ne sera pas remplie et on gardera les deux boîtes englobantes.

3.1.1.2 Fusion par chevauchement

La fusion par chevauchement est utilisé dans [Broggi *et al.*, 2007]. Cette fusion est appliquée de la même manière que la fusion par aire. Nous calculons pour chaque paire de boîtes englobantes leur taux de recouvrement (voir équation (3.2)). Si ce ratio est supérieur à 0,8 alors nous supprimons la boîte englobante qui a le plus faible score de contenir un arbre.

$$\frac{Aire(B_1 \cap B_2)}{min(Aire(B_1), Aire(B_2))} > 0,8 \tag{3.2}$$

avec B_1 et B_2 les deux boîtes englobantes.

À la différence de la fusion par aire, si une boîte englobante est incluse dans une autre boîte englobante beaucoup plus grande, il est quand même décidé de fusionner et de garder celle qui a le score le plus élevé.

3.1.1.3 Autres solutions non retenues

Nous avons également essayé deux autres approches qui au lieu de supprimer la boîte englobante avec le score le plus faible, consistent soit à prendre la moyenne des coordonnées des deux boîtes englobantes, soit à prendre la moyenne des coordonnées pondérée

par les scores d'appartenance à la classe "arbre" des deux boîtes englobantes. Les stratégies de fusion par aire et de fusion par chevauchement, c'est-à-dire celles qui sont plus agressives dans la décision de ne choisir que le résultat de la meilleure prédiction obtiennent de meilleures performances. Nous n'avons donc pas retenu les deux approches utilisant la moyenne et la moyenne pondérée des coordonnées des boîtes englobantes.

3.1.2 Algorithmes en compétition

3.1.2.1 Apprentissage automatique en deux étapes

Nous avons utilisé deux méthodes d'apprentissage automatique classiques habituellement employés dans le domaine de la télédétection : un Séparateur à Vaste Marge (SVM) et un Random Forests(RF) tous deux utilisant le descripteur HOG pour réaliser une tâche de localisation d'arbres urbains. Le SVM, le RF ainsi que le descripteur HOG sont décris dans le chapitre 2. Les SVM sont très populaires dans le domaine de la télédétection, ils ont été largement utilisés pour traiter différents problèmes de classification et de localisation. Par exemple, ils ont été utilisés pour des problèmes de détection d'objets, tels que la détection de routes [Das *et al.*, 2011] ou la détection d'avions [Sun *et al.*, 2010; Zhang *et al.*, 2014, 2015].

Que ce soit pour le SVM ou bien pour le RF, nous devons fixer les paramètres de ces algorithmes qui vont nous permettre d'obtenir les meilleures performances possibles. En ce qui concerne le RF, les deux paramètres principaux sont le nombre d'arbres de décision qui composent le RF ainsi que la profondeur de ces arbres de décision. Pour le SVM, comme nous l'avons vu dans le chapitre 2, le paramètre qui doit être fixé est le paramètre de contrainte C qui permet de contrôler le compromis entre la largeur de la marge et le nombre d'erreurs de classification. Pour fixer ces paramètres, nous avons utilisé une base de validation sur laquelle nous avons réalisé des tests itératifs. Nous avons ainsi pu choisir les paramètres permettant de maximiser les performances, c'est-à-dire la F-mesure.

En ce qui concerne l'approche par *deep learning*, nous utilisons les deux architectures de réseaux qui ont gagné le challenge ImageNet en 2012 et en 2015. Ces deux réseaux de classification sont AlexNet [Krizhevský *et al.*, 2012] et GoogLeNet [Szegedy *et al.*, 2015], nous les présentons ci-dessous.

3.1.2.2 AlexNet

AlexNet [Krizhevský *et al.*, 2012] apparaît en 2012 lors du challenge ImageNet [1]. Ce réseau a permis aux auteurs d'obtenir les meilleures performances sur la base de données ImageNet en surpassant de loin le deuxième meilleur concurrent avec un écart de 10% sur l'erreur top-5. Ce réseau (voir figure 3.3) se compose de 5 couches convolutionnelles. Chaque couche convolutionnelle est suivie d'une fonction d'activation ReLU [Nair et Hinton, 2010] et d'une opération *max-pooling*.

1. http ://www.image-net.org

Le tableau 3.1 donne une description détaillée de l'architecture AlexNet. Dans notre cas nous avons modifié cette architecture car à l'origine ce réseau prenait en entrée des images de 224×224 pixels avec les canaux RVB. Nos imagettes étant de taille 64×64 pixels, nous avons réduit le pas de la première couche de convolution à 1. De plus, puisque nous entraînons le réseau sur 2 classes, le nombre de sorties de la couche linéaire et de la couche softmax est 2.

Type	Taille du noyau / Pas	# de sorties
Convolution	11×11 / 4	96
Max pool	3×3 / 2	96
Convolution	5×5 / 1	256
Max pool	3×3 / 2	256
Convolution	3×3 / 1	384
Convolution	3×3 / 1	384
Convolution	3×3 / 1	256
Max pool	3×3 / 2	192
Fully connected		4096
Dropout(50%)		4096
Fully connected		4096
Dropout(50%)		4096
Linéaire		1000
Softmax		1000

TABLEAU 3.1 – Présentation de l'architecture AlexNet. Toutes les couches de convolution sont suivies de la fonction d'activation ReLU. Le type "Linéaire" correspond à une couche Fully connected sur laquelle nous n'appliquons pas de fonction d'activation.

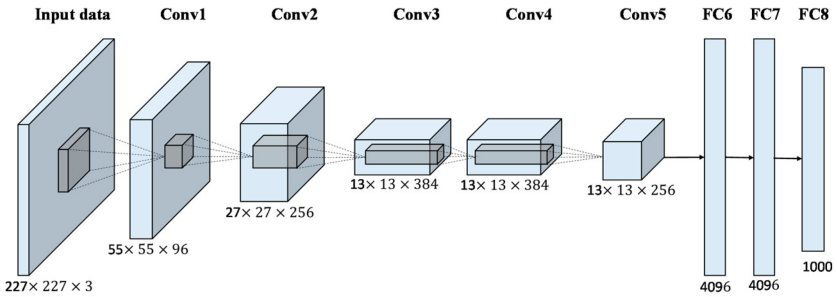


FIGURE 3.3 – Schéma du réseau AlexNet [Krizhevský *et al.*, 2012] composé de 8 couches en tout, 5 couches de convolution et de 2 couches entièrement connectées et 1 couche linéaire (on n'applique pas de fonction d'activation).

3.1.2.3 GoogLeNet

GoogLeNet [Szegedy *et al.*, 2015] est la première architecture utilisant le concept "d'inception". Le principe de ce concept est de réaliser en parallèle plusieurs couches de convo-

lution avec des noyaux de taille différente (voir figure 3.4). Cela permet au réseau de "choisir" la taille de noyau la plus appropriée au problème. L'idée principale de l'architecture inception est basée sur la découverte d'un optimum local faible dans un réseau convolutionnel qui peut être approximé facilement par le biais des convolutions mises en parallèle. Ce réseau a obtenu les meilleures performances sur la base de données ImageNet en 2015.

Le tableau 3.2 donne une description détaillée de l'architecture GoogLeNet. Nous avons aussi appliqué des modifications sur cette architecture. Cette architecture prend en entrée des images de 224 × 224 pixels avec les canaux RVB, nous avons donc réduit le pas à 1 pour la première couche de convolution et le premier *pooling max*. De plus, puisque nous entraînons le réseau sur 2 classes, le nombre de sorties de la couche linéaire et de la couche softmax est 2.

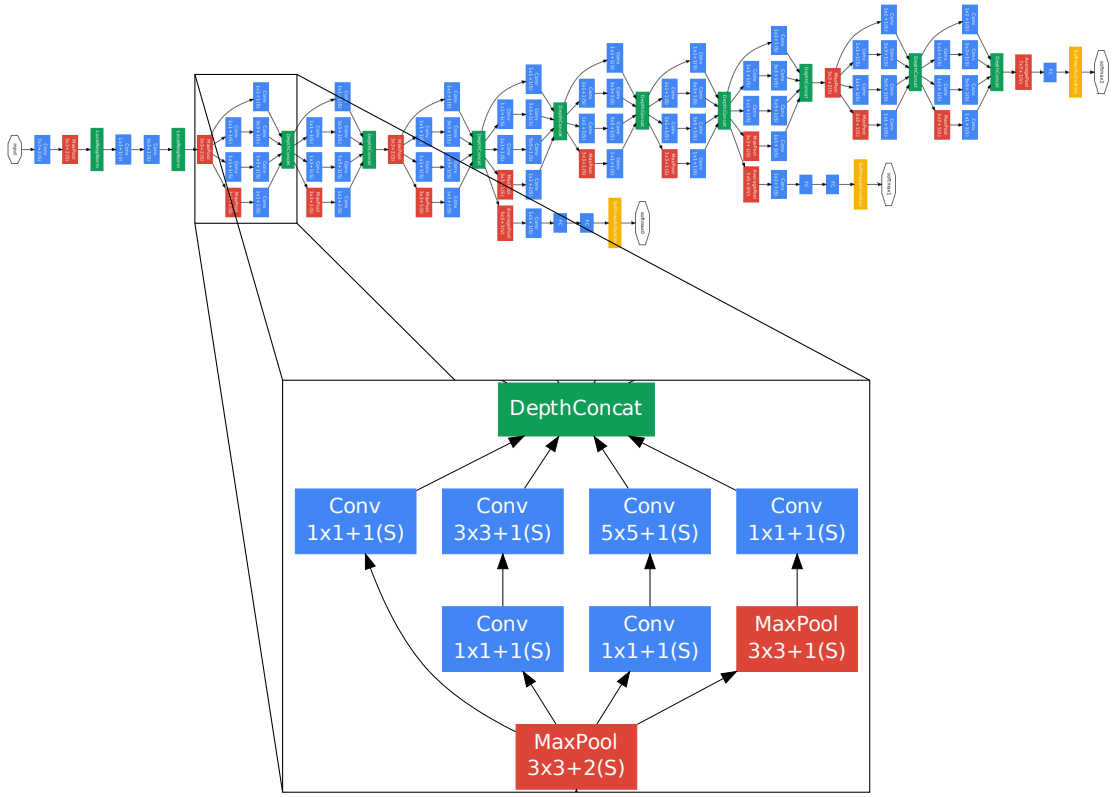


FIGURE 3.4 – Schéma du réseau GoogLeNet [Szegedy *et al.*, 2015] composé de 22 couches.

Type	Taille du noyau / Pas	Profondeur	# 1 × 1	# 3 × 3 Réduit	# 3 × 3	# 5 × 5 Réduit	# 5 × 5	# de sorties
Convolution	7 × 7 / 2	1						64
Max pool	3 × 3 / 2	0						64
Convolution	3 × 3 / 1	2		64	192			192
Max pool	3 × 3 / 2	0						192
Inception (3a)		2	64	96	128	16	32	256
Inception (3b)		2	128	128	192	32	96	480
Max pool	3 × 3 / 2	0						480
Inception (4a)		2	192	96	208	16	48	512
Inception (4b)		2	160	112	224	24	64	512
Inception (4c)		2	128	128	256	24	64	512
Inception (4d)		2	112	144	288	32	64	528
Inception (4e)		2	256	160	320	32	128	832
Max pool	3 × 3 / 2	0						832
Inception (5a)		2	256	160	320	32	128	832
Inception (5b)		2	384	192	384	48	128	1024
Avg pool	7 × 7 / 1	0						1024
Dropout (40%)		0						1024
Linéaire		1						1000
Softmax		0						1000

TABLEAU 3.2 – Présentation de l'architecture GoogLeNet. Toutes les couches de convolution, y compris dans les modules inception, sont suivies de la fonction d'activation ReLU. Les couches "# 3 × 3 Réduit" et "# 5 × 5 Réduit" correspondent au nombre de noyaux 1 × 1 qui précèdent des noyaux 3 × 3 ou 5 × 5. Le type "Linéaire" correspond à une couche Fully connected sur laquelle nous n'appliquons pas de fonction d'activation.

3.2 Évaluation des différentes méthodes de classification

Nous présentons ici le protocole que nous avons mis en place afin de valider notre approche. Dans un premier temps nous expliquons le protocole expérimental utilisé lors de nos expérimentations. Nous présentons aussi dans cette partie les mesures que nous avons employées afin de comparer les algorithmes que nous avons mis en compétition. Dans un deuxième temps nous présentons les résultats que nous avons obtenus en utilisant les différentes méthodes.

3.2.1 Évaluation du protocole et mesures

À partir de la base de données *Vaihingen* décrite dans la section 1.4, nous avons utilisé 19 images avec les canaux rouge, vert, proche infrarouge et MNS pour évaluer les performances des différentes méthodes. Toutes les expériences ont été réalisées avec une validation croisée 5 fois.

Nous avons entraîné tous les classifieurs pour séparer deux classes, la classe "arbre" et la classe "autre". Plus en détail, les imagettes pour la classe "arbre" sont obtenues par un étiquetage manuel alors que la classe "autre" est obtenue en faisant des extractions d'imagettes aléatoirement (qui ne sont pas des arbres) du jeu de données original. Afin de réaliser l'apprentissage, nous avons redimensionné toutes les imagettes à une taille fixe de 64 × 64 pixels. De plus, pour augmenter le nombre d'imagettes pour la classe "arbre", nous avons appliqué une rotation de 90°, 180° et 270° de toutes les images redimensionnées. Après augmentation, nous avons environ 5 000 imagettes "arbre" et 40 000 imagettes "autre". Nous avons entraîné nos classifieurs sur 4 500 imagettes "arbre" et 36 000 imagettes "autre" et nous nous sommes servis de 500 imagettes "arbre" et 4 000 imagettes "autre" pour constituer notre base de validation.

Notre base de test est composée d'une vingtaine d'images ne faisant pas parties des images sur lesquelles nous avons extrait nos imagettes "arbre" et "autre". Les images de la base de test sont de taille variable (de 125 × 150 pixels jusqu'à 550 × 725 pixels) et contiennent une centaine d'arbres.

Pour évaluer les résultats, nous calculons l'indice de Jaccard entre la boîte englobante détectée et la boîte correspondant à la vérité terrain. Pour déterminer si la boîte englobante est correctement placée, nous avons utilisé la même évaluation que le challenge Pascal Voc [2], cette évaluation est définie équation 3.3 [3]. Le principe de cette évaluation est de tester si l'indice de Jaccard entre la boîte englobante détectée B_d et la boîte correspondant à la vérité terrain B_{vt} est supérieur à 0,5, si c'est le cas, la méthode utilisée a correctement détecté et localisé l'arbre présent dans la B_{vt}.

2. http ://host.robots.ox.ac.uk/pascal/VOC/voc2012/

3. Attention, la notion d'union tel que définie dans le challenge Pascal Voc n'est pas une union ensembliste, mais correspond à l'aire d'une boîte recouvrant les deux boîtes englobantes considérées.

$$label = \begin{cases} 1 \quad \text{si} \quad \frac{Aire(B_d \cap B_{vt})}{Aire(B_d \cup B_{vt})} > 0,5 \\ \\ 0 \quad \text{sinon} \end{cases} \tag{3.3}$$

Une illustration de la méthode d'évaluation est présentée figure 3.5, avec en bleu la détection, en rouge la vérité terrain, en vert l'intersection entre la détection et la vérité terrain et en noir leur union. Dans les exemples présentés figure 3.5, nous présentons différentes configurations possibles. À gauche il y a le cas où les deux boîtes sont presque parfaitement superposées, dans ce cas le label sera 1. Au milieu nous avons le cas où les deux boîtes sont très éloignées et ne s'intersectent quasiment pas, dans ce cas le label sera 0. Enfin à droite nous avons le cas où la détection est incluse dans la vérité terrain mais l'indice de Jaccard est inférieur à 0,5 le label sera donc 0.

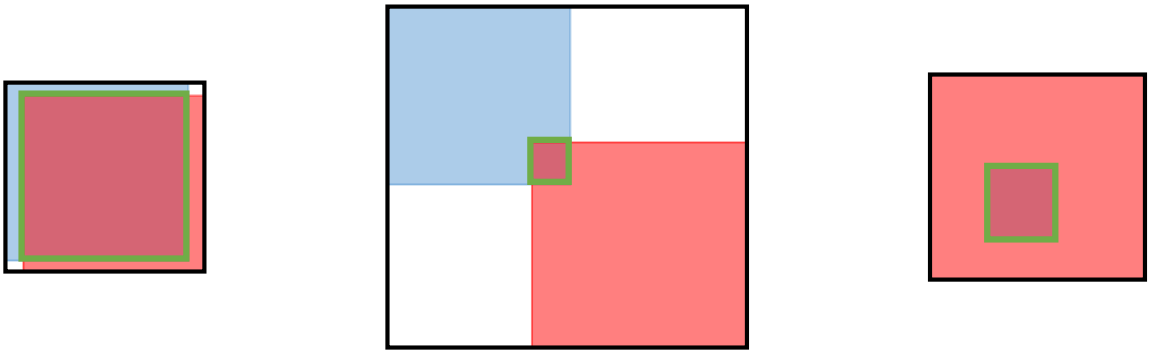


FIGURE 3.5 – Illustration de la méthode d'évaluation utilisant l'indice de Jaccard, avec en bleu la détection, en rouge la vérité terrain, en vert l'intersection des deux et en noir l'union [3] des deux. Sur la gauche nous avons le cas où deux boîtes sont presque superposée, le label sera donc 1. Au milieu nous avons le cas où l'aire de l'intersection des deux boîtes est beaucoup plus petite que celle de leur union, le label sera donc 0. Sur la droite nous avons le cas où la boîte de la détection est incluse dans la boîte de la vérité terrain. Dans le cas de cet exemple l'aire de l'union des deux boîtes étant largement supérieure à celle de leur intersection, le label sera 0.

Pour nos expériences, nous avons utilisé trois autres mesures pour évaluer les différentes méthodes : le rappel noté R présenté équation (3.4), la précision notée P présentée équation (3.5), et la F-mesure notée F présentée équation (3.6).

$$R = \frac{VP}{VP + FN} \tag{3.4}$$

$$P = \frac{VP}{VP + FP} \tag{3.5}$$

$$F = \frac{2RP}{P + R} \tag{3.6}$$

avec VP le nombre de vrais positifs, FP le nombre de faux positifs et FN le nombre de faux négatifs.

Une illustration de vrais positifs, de faux positifs et de faux négatifs est présentée figure 3.6. Les vrais positifs sont représentés en vert, il s'agit du cas où un arbre a correctement été localisé. Les faux positifs sont représentés en jaune. Les faux positifs représentent les cas où le classifieur prédit un arbre à une certaine position qui est absent dans la vérité terrain. Il peut s'agir d'un oubli de l'annotateur ou simplement d'une erreur du classifieur. Les faux négatifs sont représentés en bleu dans la figure. Nous avons un faux négatif à chaque fois qu'un arbre présent dans la vérité terrain n'a pas été détecté par le classifieur.

FIGURE 3.6 – Illustration graphique de vrais positifs, de positifs et de faux négatifs. Dans cet exemple, les vrais positifs sont lorsque l'on localise correctement un arbre, ils sont représentés en vert. Les faux positifs sont représentés en jaune, et les faux négatifs sont représentés en bleu.

3.2.2 Paramètres

Nous présentons ici les différents paramètres que nous avons utilisés avec les classifieurs et le descripteur HOG afin de réaliser l'apprentissage de notre base composée de 4 500 imagettes "arbre" et 36 000 imagettes "autre". Les paramètres des deux CNNs utilisés, AlexNet et GoogLeNet, sont présentés dans le tableau 3.3. Nous arrêtons l'apprentissage des CNNs quand ils ont convergé et avant qu'ils ne fassent du sur-apprentissage. GoogLeNet est un CNN plus gros qu'AlexNet, c'est pourquoi nous avons dû faire son apprentissage sur plus d'itérations. De plus, nous avons divisé le *learning rate* par 10 toutes les 13 333 itérations pour AlexNet et toutes les 20 000 itérations pour GoogLeNet.

Pour réaliser nos expérimentations, nous avons utilisé le descripteur HOG-cascade proposé par [Qiang *et al.*, 2006], la construction de ce descripteur est décrite section 2.1.1.4. Le résultat de ce descripteur est un vecteur de caractéristiques de dimension 680 pour

	AlexNet	GoogLeNet
Learning rate	10^{-4}	10^{-4}
Batch size	64	24
Momentum	0.9	0.9
Weight decay	0.0005	0.0005
Itérations	40 000	60 000

TABLEAU 3.3 – Paramètres que nous avons utilisés pour les CNNs AlexNet et GoogLeNet.

chaque canal, que nous donnons aux classifieurs SVM et RF. Pour le RF, nous avons fixé le nombre d'arbres aléatoires générés à 200 et nous n'avons pas imposé de limites de profondeur aux arbres aléatoires. En ce qui concerne le SVM, nous avons utilisé un noyau RBF, un gamma à 0,1 et nous avons fixé le paramètre C à 0,8. Afin de fixer les paramètres du RF et du SVM, nous avons utilisé la base de validation composée de 500 imagettes de la classe "arbre" et 4 000 imagettes de la classe "autre". Pour le RF et le SVM, nous avons utilisé l'implémentation en python fourni par la bibliothèque Scikit-learn [Pedregosa *et al.*, 2011].

3.2.3 Résultats des expérimentations

Le tableau 3.4 montre les résultats obtenus avec les différentes méthodes : *AlexNet*, *GoogLeNet*, *SVM* (avec HOG) et *RF* (avec HOG). Toutes les méthodes ont été couplées avec les deux stratégies de fusion, c'est-à-dire la fusion par aire et la fusion par chevauchement.

	AlexNet	GoogLeNet	HOG+SVM	HOG+RF
Aire				
Rappel	41.56%	**46.99%**	26.66%	38.67%
Précision	24.28%	**29.24%**	0.95%	7.77%
F-Mesure	30.41%	**35.68%**	1.83%	10.91%
Chevauchement				
Rappel	**49.28%**	48.96%	21%	33.47%
Précision	22.57%	**25.71%**	1.54%	10.47%
F-Mesure	30.63%	**33.32%**	2.88%	13.78%

TABLEAU 3.4 – Résultats obtenus sur les deux CNN et les deux méthodes d'apprentissage automatique sur lesquelles nous avons appliqué la fusion par aire et la fusion par chevauchement. Afin d'évaluer nos résultats, nous avons utilisé le rappel, la précision et la F-mesure.

Comme on peut le constater, les meilleurs résultats sont obtenus avec les CNN lorsque la fusion par aire est utilisée. Cette stratégie de fusion semble plus restrictive que celle par chevauchement. Puisque les CNN créent des vecteurs de caractéristiques avec un haut

niveau d'abstraction, cette fusion leur permet de réduire considérablement le nombre de faux positifs et donc d'avoir une meilleure précision.

FIGURE 3.7 – Exemple des résultats que nous avons obtenus : la vérité terrain est présentée sur l'image de gauche, sur l'image du milieu nous avons les résultats obtenus avec AlexNet, et enfin à droite nous avons les résultats obtenus avec le SVM+HOG.

Le *RF* et le *SVM* atteignent des performances bien inférieures à celles des réseaux de neurones convolutionnels. Contrairement aux CNN, la stratégie de fusion par chevauchement donne de meilleures performances que la fusion par aire. Les performances obtenues avec ces méthodes sont extrêmement faibles. Cela peut être dû au fait que les arbres sont souvent très proches les uns des autres, ce qui les rend difficiles à différencier sur la base de leur bordure (voir Figure 3.7) qui devrait être soulignée par les descripteurs *HOG*.

Les résultats de détection sont faibles et l'on peut se demander si cela est dû à la localisation et à l'aspect multi-résolutions ou bien si cela est dû aux méthodes utilisées. Nous rapportons ici les résultats obtenus lors de l'évaluation individuelle des classifieurs d'imagettes sans intégrer un mécanisme de localisation avec fenêtre glissante.

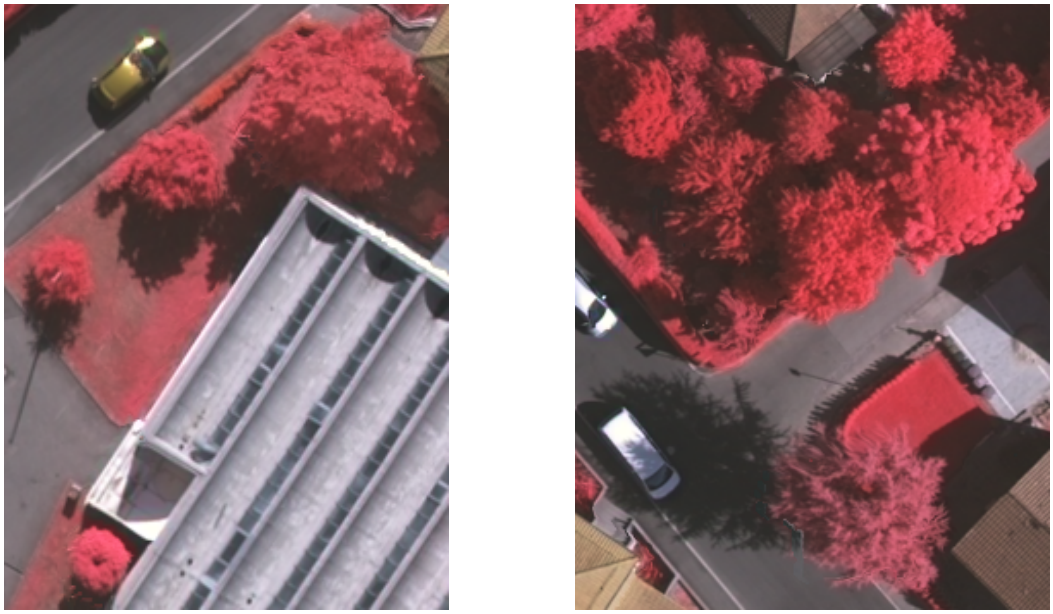
	AlexNet	GoogLeNet	HOG+SVM	HOG+RF
Rappel	97.80%	**98.05%**	52.81%	54.79%

TABLEAU 3.5 – Rappel pour les différentes approches.

Le tableau 3.5 montre le rappel des différentes approches. Pour cette expérience, nous avons comparé les performances des classifieurs sans tenir compte du problème de localisation. À cette fin, nous n'appliquons aucune stratégie de fusion et nous nous concentrons uniquement sur le rappel obtenu par les différentes méthodes de classification. L'objectif est de déterminer si les classifieurs sont capables de trouver tous les arbres dans l'ensemble des images de test sans nous préoccuper des boîtes englobantes redondantes et de la fusion, c'est pourquoi nous ne nous intéressons pas à la précision dans cette expérience mais uniquement au rappel.

Toujours dans cette expérience, on peut noter que les CNN atteignent des performances proches de 100% alors que les techniques classiques de classification par télédétection sont à peine au-dessus des 50%. Les deux architectures de réseau détectent plus de 97% des arbres présents dans la base de test contre 52% pour les approches d'apprentissage automatique en deux étapes. Cette expérience permet de montrer la difficulté de la tâche que l'on souhaite effectuer. On peut constater en regardant que le problème est ici lié à la localisation. En effet, nous sommes capables de classifier correctement plus de 97% des arbres, cependant la phase de localisation fait chuter drastiquement les performances. En effet, on peut noter que lorsque l'on utilise un mécanisme de localisation, le rappel peine à atteindre les 50% même lorsqu'un CNN est utilisé.

Ce phénomène peut s'expliquer par le fait que la tâche de localisation est plus difficile que la tâche de classification. Lors de la classification les arbres sont centrés dans les imagettes, dans le cas de la localisation il peut y avoir plusieurs arbres qui se chevauchent et il devient difficile pour les classifieurs de déterminer la position de chaque arbre. Un exemple de la difficulté de la tâche de localisation est illustré figure 3.8. Dans cet exemple, nous montrons deux cas différents. Pour le premier cas, c'est-à-dire l'image (a) de la figure, il s'agit d'un exemple où la localisation est facile. En effet, on peut voir sur l'image (a) que les arbres sont bien séparés et visibles, et leurs contours sont nets. Le second cas, c'est-à-dire l'image (b) de la figure, nous montrons un cas où la tâche de localisation est difficile. Les arbres sont collés les uns aux autres et se chevauchent, il est très difficile de les distinguer.



(a) (b)

FIGURE 3.8 – Illustration de la difficulté de la localisation. L'image (a) montre un cas où la localisation est simple, c'est-à-dire le cas où les arbres sont bien visibles et séparés les uns des autres. L'image (b) montre un cas où la localisation est difficile. Les arbres sont collés, ils se chevauchent, même pour un humain il est difficile de distinguer tous les arbres.

Nous avons aussi remarqué durant nos expérimentations qu'il y avait des erreurs d'annotation. La figure 3.9 représente ce cas. La boîte orange représente un arbre qui a été oublié par l'annotateur et les boîtes bleues représentent les arbres qui ont été annotés. Ces erreurs d'annotations vont faire augmenter le nombre de faux positifs et ainsi faire baisser la précision.

FIGURE 3.9 – Exemple d'image sur laquelle il y a une erreur d'annotation. Les boîtes bleues représentent les arbres qui ont été annotés et la boîte orange représente un arbre qui a été oublié par l'annotateur.

Un autre phénomène fait drastiquement baisser la précision, il s'agit de la détection des haies. En effet, durant nos expérimentations nous avons pu observer qu'il y a une confusion avec les haies. Les classifieurs ont tendance à classer les haies comme étant des arbres, cela fait donc augmenter le nombre de faux positifs et donc fait baisser la précision.

La figure 3.10 présente un récapitulatif des résultats que nous avons obtenus avec les deux approches d'apprentissage automatique classique en deux étapes et avec les deux réseaux de neurones.

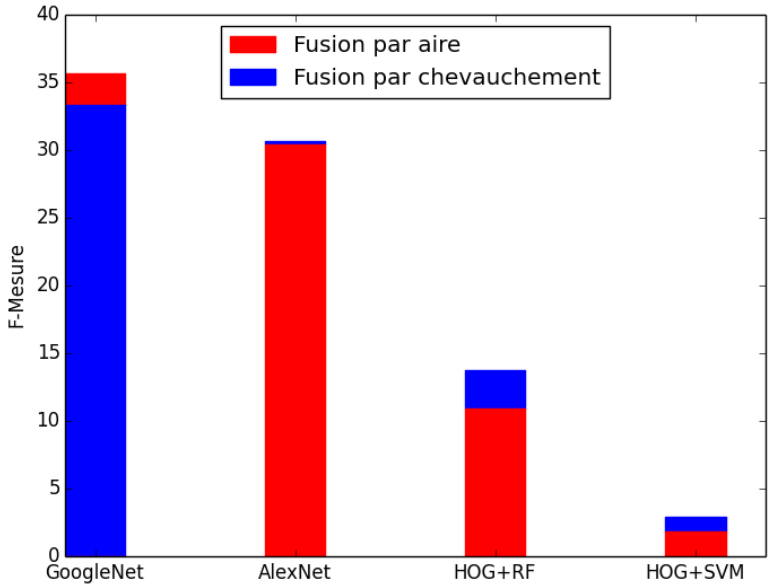


FIGURE 3.10 – F-Mesures obtenues sur les expériences réalisées sur la tâche de localisation. En rouge ce sont les résultats que nous avons obtenus en appliquant la fusion par aire et en bleu ce sont les résultats obtenus avec la fusion par chevauchement.

3.3 Conclusions et discussion

En conclusion de ce chapitre, nous pouvons observer que les approches CNN sont bien adaptées à la tâche de détection des arbres urbains dans les données aériennes multi-sources. Nous avons montré lors de notre première expérimentation que les réseaux de neurones convolutionnels donnent de meilleures performances sur la tâche de localisation d'arbres urbains que les approches d'apprentissage automatique classique en deux étapes (voir figure 3.10). Suite aux résultats de cette expérimentation, nous allons continuer d'utiliser les CNN dans la suite de nos travaux.

Nos résultats montrent que le descripteur *HOG* [Qiang *et al.*, 2006] n'est pas suffisant pour détecter et localiser les arbres urbains sur des images aériennes multi-sources. Comme on peut le voir sur la figure 3.8, les arbres peuvent avoir des formes et des tailles très variables. Les expériences que nous avons menées nous ont montré que le descripteur HOG [Qiang *et al.*, 2006] n'offre pas une représentation suffisamment robuste pour faire face à cette hétérogénéité. De plus, le fait que les arbres puissent se chevaucher entre eux ou bien avec d'autres objets, le descripteur *HOG* ne permet pas de différencier leur frontière clairement.

Un autre inconvénient lié à l'utilisation d'une méthode d'apprentissage automatique classique en deux étapes est que les descripteurs qui peuvent être efficaces pour un objet urbain d'un certain type peuvent être inutiles pour un objet urbain d'un autre type. Choisir le bon ensemble de descripteurs pour un objet particulier implique des connaissances et une expérience humaine qui ne sont pas toujours disponibles.

Dans notre cas, nous avons vu que les approches CNN permettent d'obtenir les meilleures performances sur la localisation des arbres urbains à partir de données aériennes multi-sources. De nombreuses études récentes [Fang *et al.*, 2016; Liu *et al.*, 2016; Maninis *et al.*, 2018] ont montré que les CNN fournissent automatiquement des caractéristiques de haut niveau, robustes [LeCun *et al.*, 1995] et bien adaptées à la classification [Krizhevský *et al.*, 2012; Szegedy *et al.*, 2015; Wagstaff *et al.*, 2018] et la localisation [Zhang *et al.*, 2018; Yu *et al.*, 2018].

Notre deuxième expérimentation nous a permis de souligner que la difficulté réside dans la localisation et non pas la classification. Nous avons pu constater que lors de la phase de localisation, les performances de toutes les méthodes diminuent. Nous avons identifié différents éléments qui pourraient être la cause de la chute des performances lors du passage de la classification à la localisation.

Chapitre 4

Fusion des données

Sommaire

4.1 Introduction

Comme nous l'avons vu dans le chapitre précédent, les approches par *deep learning* permettent d'obtenir de meilleurs résultats que les méthodes d'apprentissage automatique en deux étapes. C'est pour cette raison que nous avons choisi de poursuivre nos travaux avec les CNNs.

Nous présentons dans ce chapitre une première contribution où nous avons étudié deux façons de combiner des données multi-sources. Le deuxième objectif de cette étude est de montrer l'intérêt d'utiliser des données multi-sources. En effet, nous défendons dans ce chapitre, la thèse selon laquelle l'utilisation de plusieurs sources de données a un intérêt puisqu'une utilisation intelligente permet d'augmenter les résultats de classification et de détection.

Cependant, l'utilisation de données multi-sources est complexe car chacune des sources peut nous fournir des informations complémentaires mais de nature différente et à des échelles différentes [Schmitt et Zhu, 2016]. C'est pour cette raison qu'il est crucial de considérer le problème de l'intégration de ces informations hétérogènes dès la conception de la méthode de détection d'objets. La manière dont ces différentes données sont combinées peut avoir un impact important sur le résultat final.

Nous souhaitons souligner l'importance du pré-traitement des images lors de l'utilisation d'une approche *deep learning*, notamment avec la préparation et le pré-traitement des données d'apprentissage qui sont d'une importance primordiale sur les performances des CNN.

4.2 Localisation des objets et fusion des résultats

Afin de localiser les arbres sur les images testées, nous avons utilisé la même approche que dans le chapitre 3, c'est-à-dire une fenêtre glissante multi-résolution. Les couronnes des arbres ne faisant pas toutes le même diamètre, cette méthode nous permet de détecter tous les arbres indépendamment de leur taille.

Comme les imagettes extraites de la fenêtre glissante doivent être de la même taille que les imagettes utilisées lors de la phase d'apprentissage, au lieu de faire varier la taille de la fenêtre glissante, nous avons fait varier la taille des images testées. Nous avons redimensionné chaque image de 30% jusqu'à 300% de leur taille d'origine avec un pas de 10%, on doit appliquer notre fenêtre glissante sur 28 images. La fenêtre glissante balaye les images testées à toutes les résolutions.

Comme dans le chapitre 3, l'application de la fenêtre glissante multi-résolution va entraîner une accumulation de boîtes englobantes sur certaines zones. Nous avons donc ici aussi appliqué une stratégie de fusion. Comme nous avons pu le constater dans le chapitre précédent, la fusion par chevauchement a permis d'obtenir les meilleures performances, c'est-à-dire la meilleure F-mesure avec AlexNet. Étant donné que nous avons choisi de n'utiliser que AlexNet pour cette contribution, nous avons choisi de n'utiliser

qu'une seule fusion, la fusion par chevauchement.

Pour rappel, la fusion par chevauchement va comparer toutes les paires de boîtes englobantes. Pour chaque paire, nous calculons si le chevauchement des deux boîtes englobantes a un pourcentage supérieur à 80% (voir équation (4.1)). Si c'est le cas, la boîte englobante ayant la plus faible probabilité de contenir un arbre est supprimée.

$$\frac{Area(B1 \cap B2)}{min(Area(B1), Area(B2))} > 0,8 \tag{4.1}$$

avec B1 et B2 deux boîtes englobantes.

Pour rappel, nous travaillons ici avec des images optiques, avec les canaux R et V, mais aussi avec du Pir (proche infrarouge) et du MNS.

Lorsque l'on travaille sur des image contenant de la végétation, il peut-être intéressant d'utiliser des indices de végétation. Ces indices permettent de discriminer la végétation du reste de l'image, mais aussi de décrire l'état d'un phénomène comme rendre compte du stade de croissance végétale à un moment donné par exemple. De plus, ils peuvent aussi nous permettre de détecter les zones d'eau très facilement.

Les indices de végétation les plus simples se basent uniquement sur des opérations entre deux bandes spectrales. Généralement, c'est le proche-infrarouge ainsi que le rouge qui sont utilisés.

Un des plus vieux indices est l'indice différentiel de végétation (DVI). Cet indice calcule simplement la différence entre la bande du proche-infrarouge et celle du rouge [Bacour *et al.*, 2006]. Cet indice est présenté équation 4.2, où ρ_{PIR} est la réflectance dans la bande proche-infrarouge et ρ_R la réflectance dans la bande rouge :

$$DVI = \rho_{PIR} - \rho_R \tag{4.2}$$

Un autre indice, appelé indice de végétation par quotient (RVI), s'exprime également de façon très simple. Il se calcule en faisant le rapport entre les bandes du PIR et du rouge [Kriegler *et al.*, 1969; Jordan, 1969].

$$RVI = \frac{\rho_{PIR}}{\rho_R} \tag{4.3}$$

L'inconvénient de ces deux indices est qu'ils sont sensibles aux variations atmosphériques et à la contribution spectrale des sols. De plus, les valeurs de l'indice RVI saturent sur les zones de végétation très denses car la réflectance dans la bande rouge devient alors très faible.

L'indice le plus connu et le plus utilisé est l'indice de végétation par différence normalisée (NDVI) [Rouse Jr *et al.*, 1974; Tucker, 1979]. Cet indice est défini comme suit :

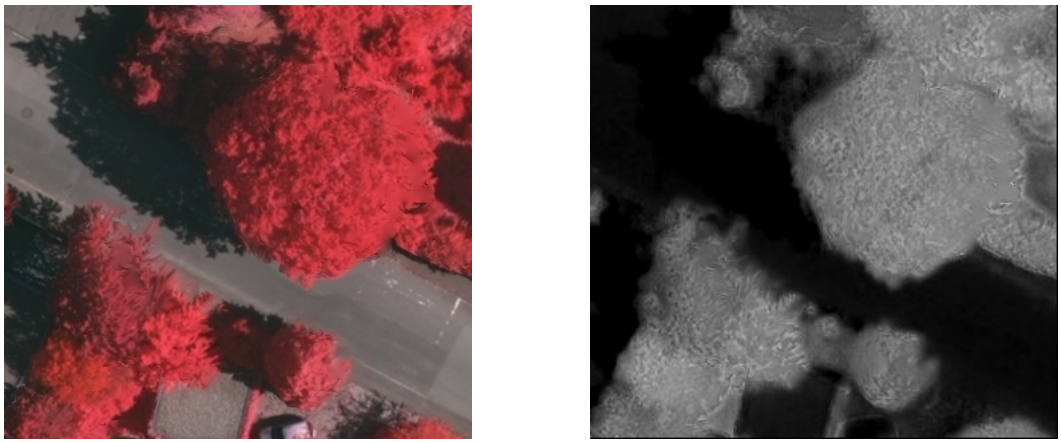
$$NDVI = \frac{\rho_{PIR} - \rho_R}{\rho_{PIR} + \rho_R} \tag{4.4}$$

Le fait de normaliser par la somme des deux bandes permet de réduire les effets d'éclairement. Contrairement à l'indice DVI qui est extrêmement sensible aux variations d'éclairement, le NDVI va lui conserver une valeur constante quelque soit l'éclairement global.

Les valeurs du NDVI sont comprises en théorie entre -1 et +1. Lorsque la réflectance du rouge et supérieure à celle du proche-infrarouge, c'est-à-dire lorsqu'il y a de la neige, des nuages ou bien de l'eau, le NDVI donnera des valeurs négatives. Quand le rouge et le proche-infrarouge auront des valeurs proches, ce qui est le cas pour les sols nus, le NDVI donnera des valeurs proches de zéro. Enfin, sur les formations végétales, le NDVI donnera des valeurs positives qui seront généralement entre 0,1 et 0,7. Pour les zones très denses, les valeurs du NDVI seront proches de 1.

Cet indice de végétation est largement utilisé dans le domaine de la télédétection [Meyer, 2011; Alonzo *et al.*, 2014; Madonsela *et al.*, 2018], c'est pourquoi nous avons choisi de l'utiliser.

Une illustration du NDVI est présentée figure 4.1. Nous pouvons observer sur la figure 4.1 que le NDVI nous permet d'enlever des zones qui ne sont pas de la végétation. Ainsi, nous pouvons facilement diminuer le nombre de faux positifs. En effet, les différents objets qui ont une forme similaire à celle d'un arbre et qui ne sont pas de la végétation ne sont plus présents sur le NDVI.



(a) (b)

FIGURE 4.1 – Génération des images NDVI à partir de la base de données Vaihingen : (a) Image avec les canaux proche infrarouge, rouge et vert, (b) image NDVI générée.

4.3 Combinaison des différentes sources

Le problème de la fusion de données multi-sources est un problème qui a déjà été étudié. En effet, en 2005 [Snoek *et al.*, 2005] a déjà comparé les approches ***Early Fusion*** [Snoek *et al.*, 2004] et ***Late Fusion*** [Westerveld *et al.*, 2003; Wu *et al.*, 2004; Morvant *et al.*, 2014; Zhu et Yin, 2018] en utilisant de l'apprentissage automatique en deux étapes. Comme les auteurs le décrivent dans [Snoek *et al.*, 2005], l'approche ***Early Fusion*** consiste à combiner les informations en amont, c'est-à-dire avant de les donner au classifieur. Dans le cas d'une approche par apprentissage automatique en deux étapes, l'idée est de concaténer les vecteurs de caractéristiques extraits sur toutes les sources et de donner le

vecteur issu de la concaténation à un classifieur. On peut dire que cette approche consiste à réaliser un apprentissage global sur l'ensemble des informations en une seule fois.

Pour le cas de l'approche ***Late Fusion***, dans un premier temps un apprentissage sur chacune des informations est réalisé, et dans un deuxième temps on va chercher à combiner les scores de tous les classifieurs. On peut voir dans cette approche que le but est d'essayer de diviser le travail, en ne se concentrant que sur une source à la fois et en combinant les différents scores obtenus pour chaque source afin d'obtenir un score global.

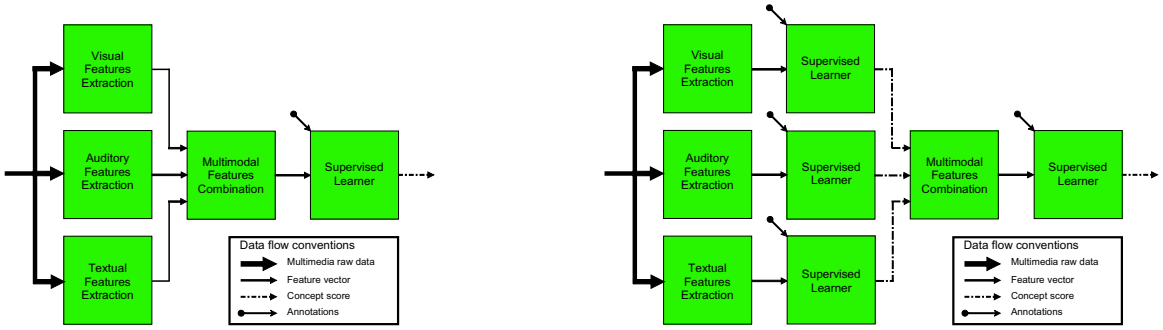


FIGURE 4.2 – Représentation de l'approche ***Early Fusion*** (gauche) et de l'approche ***Late Fusion*** (droite). Dans le cas de l'approche ***Early Fusion***, on peut voir sur le schéma de gauche que les caractéristiques extraites sur chacune des sources sont combinées et données directement à un classifieur. Dans le cas de l'approche ***Late Fusion*** à droite sur le schéma, on voit que juste après l'extraction des caractéristiques il y a une phase d'apprentissage qui est réalisée sur chacun des vecteurs de caractéristiques de manière indépendante. Ce sont les sorties des différents classifieurs qui vont être combinées et données à un autre classifieur. Source : [Snoek *et al.*, 2005].

En 2016, les auteurs de [Ergun *et al.*, 2016] proposent d'appliquer les approches ***Early Fusion*** et ***Late Fusion*** en utilisant des CNN. Les auteurs proposent notamment de fusionner les caractéristiques extraites à partir de différents CNN.

4.3.1 Early Fusion

La première façon d'intégrer des données provenant de différentes sources est une méthode "naïve" qui consiste à considérer les composantes rouge, vert, proche infrarouge et MNS comme les composantes d'une image. Nous avons appliqué cette méthode dans nos expérimentations du chapitre 3. Nous appelons ce type d'architecture Fusion prématurée ou ***Early Fusion*** (EF). Nous avons testé en utilisant les composantes rouge, vert, proche infrarouge et MNS comme composantes d'une image mais aussi en utilisant les composantes NDVI et MNS. La figure 4.3 représente le fonctionnement de cette approche. Dans le cadre bleu nous avons la combinaison des différentes sources, cela signifie que dans notre cas, nous allons créer une nouvelle image avec les composantes proche infrarouge, rouge, vert et MNS.

Nous rappelons que nous utilisons ici le CNN AlexNet. L'architecture globale de ce CNN est présentée figure 3.3, et une description détaillée est présentée tableau 3.1.

L'idée dans le cas de cette approche "naïve" est de donner l'ensemble des informations directement au réseau.

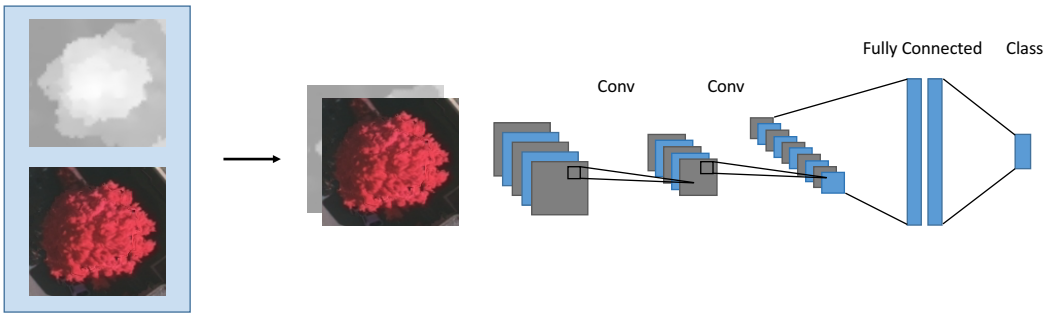


FIGURE 4.3 – Représentation de l'approche ***Early Fusion***. Dans le cadre bleu nous avons les différentes sources que nous voulons combiner, c'est-à-dire le MNS, le Pir et une image optique avec les canaux R et V, et en sortie nous avons une image de 64×64 pixels, avec les composantes proche infrarouge, rouge, vert et MNS qui est donnée au réseau.

4.3.2 Late Fusion

Nous présentons dans cette section la deuxième architecture que nous avons utilisée. Cette architecture CNN est inspirée du modèle d'un Ensemble de CNN [Xu *et al.*, 2016; Huang et Kong, 2016]. L'idée principale d'un Ensemble de CNN est de diviser l'apprentissage en plusieurs sous-modèles, c'est-à-dire en plusieurs CNN. Chaque sous-modèle va apprendre sur une partie de la base d'apprentissage, et pendant le test, nous allons fusionner les résultats obtenus avec tous les sous-modèles. Cet Ensemble de CNN nous a conduit à l'idée que nous devrions traiter les différentes informations indépendamment comme dans [Wagner *et al.*, 2016]. Nous utilisons l'architecture présentée Figure 4.4, nous appelons ce type d'architecture fusion tardive ou ***Late Fusion*** (LF).

Pour construire cette architecture, nous avons utilisé AlexNet [Krizhevský *et al.*, 2012]. Nous avons dupliqué les couches convolutives, et nous avons concaténé les deux branches, c'est-à-dire la branche qui prend en entrée le MNS et celle qui prend l'image optique ainsi que le Pir avant les couches entièrement connectées. On peut observer à partir de la figure 4.4 que le réseau a deux entrées. Chaque entrée correspond à un type de données différent. Nous pensons que ce type d'architecture permet au réseau d'extraire le meilleur descripteur pour chaque type de données et d'augmenter ainsi la F-mesure, le rappel et la précision. En effet, nous pensons que si une branche ne doit se concentrer que sur une seule source, ou bien des sources avec des valeurs proches comme avec l'image optique et le Pir, elle pourra extraire des caractéristiques qui sont propres aux sources et ainsi des caractéristiques qui seront plus discriminantes.

Durant la construction de ce réseau nous avons fait varier le nombre de paramètres. Dans un premier temps nous avons divisé par 2 les paramètres dans chacune des branches afin d'obtenir le même nombre de paramètres que dans un AlexNet classique (voir le tableau 3.1). Nous avons par la suite laissé les paramètres de AlexNet dans chacune des

branches c'est-à-dire nous avons laissé les mêmes paramètres pour les couches de convolution que dans le tableau 3.1. Après avoir réalisé quelques tests sur une sous-partie de la base de test, nous avons constaté que laisser les paramètres de AlexNet dans chacune des branches permettait d'obtenir les meilleurs résultats, nous avons donc utilisé le réseau utilisant les paramètres de AlexNet dans chacune des branches pour réaliser nos expériences.

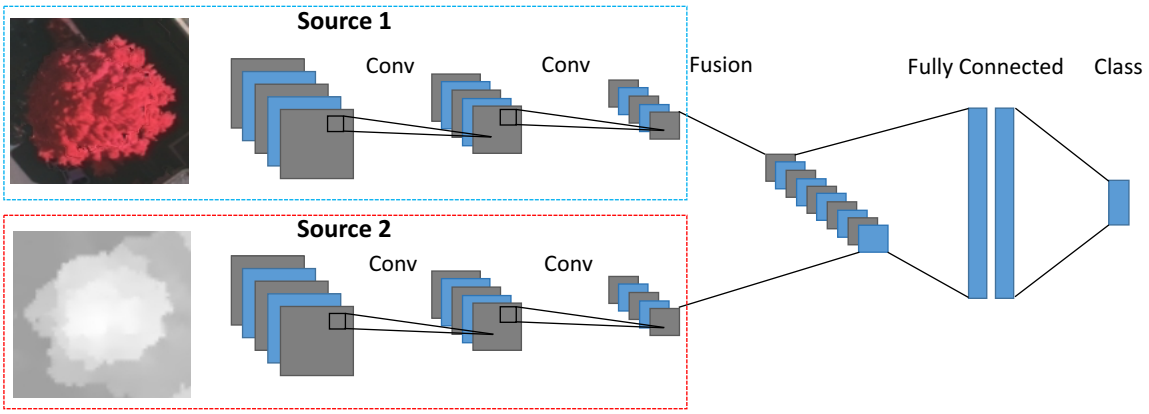


FIGURE 4.4 – Représentation de l'architecture ***Late Fusion*** avec dans le cadre bleu la première branche du réseau traitant l'image optique R,V ainsi que le Pir, et dans le cadre rouge la deuxième branche qui traite le MNS.

4.4 Évaluation expérimentale

Pour apprendre nos modèles, nous avons utilisé la même base d'apprentissage que dans le chapitre 3. Pour rappel, dans cette base nous avons annoté manuellement les arbres sur 19 images. Pour ces expériences, nous avons annoté manuellement 330 arbres en plus, ce qui nous fait un total de 1 500 arbres annotés. Après avoir extrait les 1 500 imagettes de la classe "arbre", puis nous avons appliqué 3 rotations de 90°, 180° et 270°. Nous avons donc au final 6 000 imagettes de la classe "arbre" dans notre base d'apprentissage. Pour la classe "autre", nous avons extrait 40 000 imagettes (qui ne sont pas des arbres) de façon aléatoire sur les 19 images sur lesquelles nous avons annoté nos arbres.

Notre base de test est composée d'une vingtaine d'images ne faisant pas partie des images sur lesquelles nous avons extrait nos imagettes "arbre" et "autre". Les images de la base de test sont de taille variable (de 125 × 150 pixels jusqu'à 550 × 725 pixels) et contiennent une centaine d'arbres.

Pour les expériences de ce chapitre, nous avons décidé d'utiliser un pas de déplacement plus important de la fenêtre glissante en utilisant un pas de 8 pixels contre seulement 2 pixels dans le chapitre précédent. Chaque imagette extraite par la fenêtre glissante est donnée au réseau qui nous donne un score sur la présence d'un arbre dans l'imagette. Le réseau nous permet de connaître les zones dans l'image qui ont un score élevé de contenir un arbre. Afin de déterminer si on doit garder ou non une boîte englobante

nous avons utilisé un seuil que nous avons fixé en utilisant une base de validation. Pour régler ce seuil, nous avons choisi de prendre le seuil qui maximise la F-mesure sur notre base de validation.

Pour évaluer les résultats, nous avons utilisé les mêmes métriques que dans le chapitre précédent. Pour déterminer si une boîte englobante est bien placée, nous calculons le rapport de recouvrement entre la boîte englobante détectée et la vérité terrain (voir équation (3.3)). Ensuite nous avons calculé le rappel (voir équation (3.4)), la précision (voir équation (3.5)) ainsi que la F-mesure (voir équation (3.6)).

Nous avons d'abord testé une classification sur une seule source. Ces tests nous permettent de déterminer la source la plus intéressante, c'est-à-dire celle qui donne les meilleures performances en termes de F-mesure, de rappel et de précision lorsqu'elle est utilisée seule. Ensuite, nous comparons différentes façons d'intégrer des données de différentes sources dans un CNN.

Pour valider notre expérimentation, nous effectuons une validation croisée sur notre base.

Le tableau 4.1, le tableau 4.2 et le tableau 4.3 présentent les résultats que nous avons obtenus sur la base de données Vaihingen. Ils représentent les valeurs moyennes du *Rappel*, de la *Précision* ainsi que la *F-Mesure*$_{max}$ pour les différents modèles. Pour obtenir notre courbe rappel/précision, on fixe un seuil pour lequel on considère que l'image contient un arbre. Par exemple, si on prend un seuil à 0,9, si le réseau nous dit que l'image contient un arbre avec un score de 0,8 alors on considérera que l'image ne contient pas d'arbre. Nous avons choisi de prendre le point de la courbe de rappel/précision où la F-mesure est la plus élevée.

4.4.1 Limites de l'utilisation d'une seule source

La table 4.1 montre les résultats obtenus en utilisant une seule source, ou bien en combinant l'image optique avec le Pir. Comme on peut le voir les résultats entre les différents tests sont très proches. Le NDVI nous permet d'obtenir les meilleures performances en termes de *Rappel* et de *F-Mesure*$_{max}$. La meilleure *Précision* est obtenue en utilisant le MNS. On peut observer que les meilleurs résultats en termes de *F-Mesure*$_{max}$ sont obtenus lorsque nous utilisons le NDVI.

Source	PirRV	MNS	NDVI
F-Mesure$_{max}$	60.45%	62.47%	**63.97%**
Rappel	57.89%	57.62%	**62.34%**
Précision	63.44%	**68.56%**	67.04%

TABLEAU 4.1 – Résultats en utilisant une seule source.

4.4.2 Quel est le meilleur processus de fusion ?

Le tableau 4.2 montre les résultats obtenus en utilisant la méthode ***Early Fusion***. Ici, nous pouvons voir que les meilleurs résultats sont obtenus en utilisant le NDVI et le MNS, nous obtenons une $F\text{-}Mesure_{max}$ de 75%. Cependant, lorsque nous utilisons le PirRV, les résultats sont inférieurs (67%).

EF	PirRV/MNS	NDVI/MNS
F-Mesure$_{max}$	67.12%	**75.30%**
Rappel	65.40%	**68.37%**
Précision	69.54%	**84.11%**

TABLEAU 4.2 – Résultats utilisant des données multi-sources et l'approche ***Early Fusion***. Nous avons ici les résultats des expériences menées en combinant les composantes rouge, vert, proche infrarouge et MNS, ainsi que les composantes NDVI et MNS. Les résultats sont exprimés par rapport au Rappel, à la Précision et à la F-Mesure.

Le tableau 4.3 montre les résultats que nous avons obtenus en utilisant l'architecture ***Late Fusion*** décrite dans la section 4.3.2. La conclusion reste la même que dans les expériences précédentes, les meilleurs résultats sont obtenus lorsque l'on utilise le NDVI et le MNS. En utilisant le NDVI et le MNS, nous obtenons un $F\text{-}Mesure_{max}$ de 72% contre 62% en utilisant PirRV et MNS. On peut également noter que cette architecture donne des performances moindres par rapport à l'architecture ***Early Fusion***.

LF	PirRV/MNS	NDVI/MNS
F-Mesure$_{max}$	62.14%	**72.57%**
Rappel	62.54%	**70.99%**
Précision	62.65%	**74.83%**

TABLEAU 4.3 – Résultats obtenus en utilisant des données multi-sources et l'architecture ***Late Fusion***. Nous avons ici les résultats des expériences menées en utilisant les composantes rouge, vert, proche infrarouge et MNS, ainsi que les composantes NDVI et MNS. Les résultats sont exprimés par rapport au Rappel, à la Précision et à la F-Mesure.

La figure 4.5 présente un récapitulatif des résultats que nous avons obtenus en utilisant une seule source, l'approche ***Early Fusion*** et l'architecture ***Late Fusion***.

4.4.3 Comment sélectionner les sources à fusionner ?

Nous avons calculé la corrélation entre chaque source en utilisant l'index de Jaccard. Nous calculons l'intersection des arbres trouvés dans deux sources sur l'union des arbres trouvés dans les deux sources, il s'agit de la même formule que l'équation (3.3). Les résultats sont présentés à la première ligne du tableau 4.4. La deuxième ligne du tableau 4.4

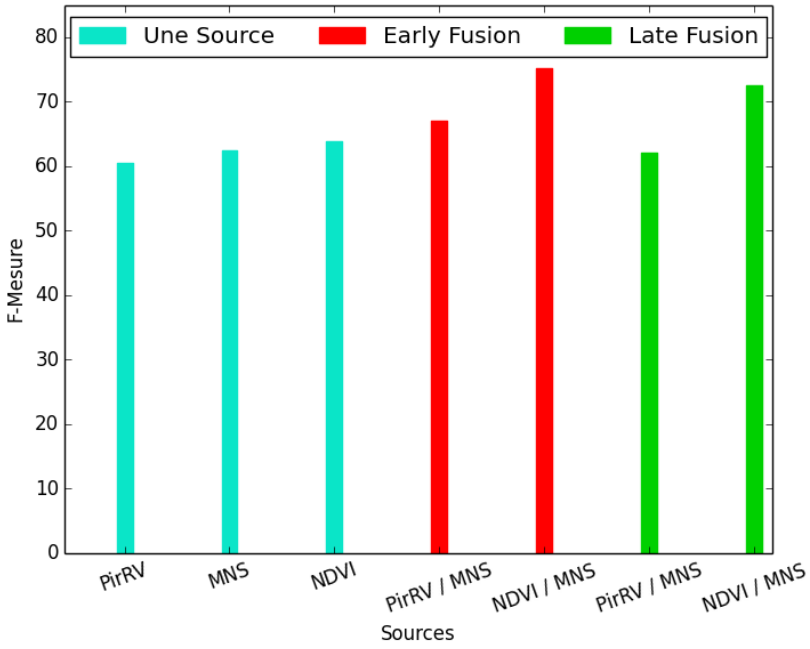


FIGURE 4.5 – Schéma récapitulatif des résultats que nous avons obtenus durant nos expérimentations en terme de F-mesure. En bleu, nous avons les résultats obtenus en utilisant durant l'apprentissage une source, ou bien la combinaison des images optiques et le Pir. Les résultats en rouge représentent les F-mesures que nous avons obtenues avec l'approche ***Early Fusion*** en combinant le Pir et le MNS ainsi que le NDVI et le MNS. Enfin, nous avons en vert les résultats que nous avons obtenus avec l'approche ***Late Fusion*** en combinant le Pir et le MNS ainsi que le NDVI et le MNS.

représente la distribution des arbres détectés seulement en utilisant une des 2 sources. L'idée de cette expérience est de montrer l'apport des différentes sources sur la tâche de localisation d'arbres urbains.

Sources	PirRV/MNS		NDVI/MNS	
Corrélation	47.86%		48.96%	
Distribution	26.47%	25.66%	28.75%	22.27%

TABLEAU 4.4 – Résultats de la corrélation entre chaque source.

Comme on peut le constater en regardant le tableau 4.4, moins de 50% des arbres détectés sont corrélés entre les sources PirRV et MNS, et les sources NDVI et MNS. Cela signifie que moins de 50% des arbres détectés sont détectés dans les deux sources. De plus, on peut remarquer que les arbres restants sont répartis équitablement entre les deux sources PirRV/MNS ou NDVI/MNS. Ces résultats ne font que confirmer les résultats présentés précédemment ainsi que l'intérêt de l'utilisation de données multi-sources.

4.5 Discussions

Comme on peut le constater sur les tableaux 4.2 et 4.3, lorsque nous utilisons l'architecture ***Late Fusion***, les résultats sont toujours inférieurs à l'approche ***Early Fusion***. En effet, lorsque nous utilisons PirRV combiné avec le MNS, les résultats passent de 62% avec l'architecture ***Late Fusion*** à 67% avec l'architecture ***Early Fusion***. De même, lorsqu'ils sont testés avec le NDVI et le MNS, les résultats passent de 72% avec l'architecture ***Late Fusion*** à 75% en utilisant l'architecture ***Early Fusion***.

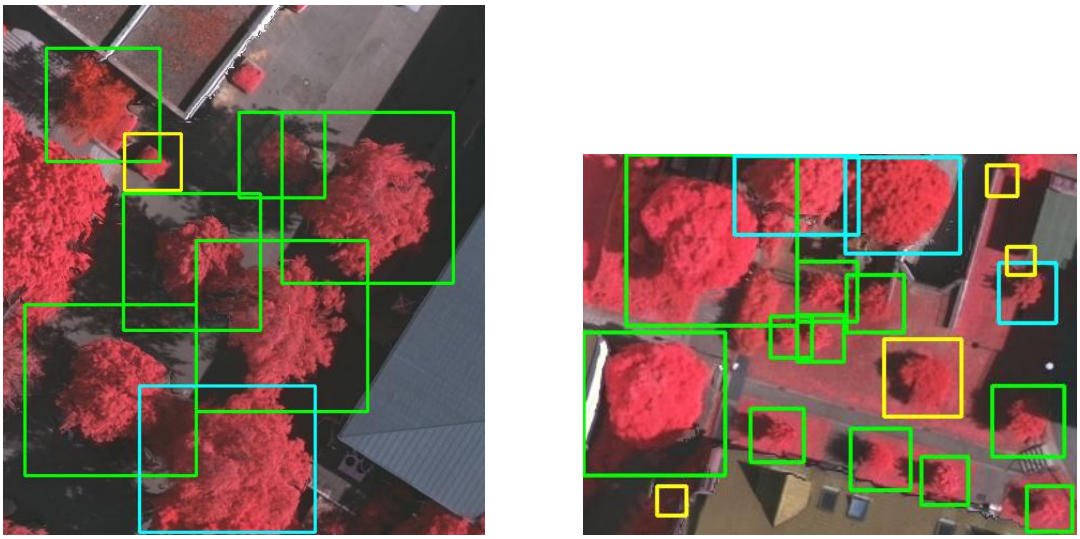
En ce qui concerne les données, on peut noter que lorsqu'on utilise PirRV seul, le résultat n'est pas si éloigné de ceux du NDVI ou du MNS. Cependant, lorsque nous combinons PirRV et MNS, le gain n'est pas significatif. En effet, comme ces données sont très différentes (leur échelle de valeur par exemple), il est très difficile pour le CNN de discriminer correctement les arbres en combinant ces deux types de données. Une solution permettant de faciliter le travail du CNN serait de normaliser les données ne pré-traitement.

De plus, nous pouvons observer que la combinaison du MNS avec le NDVI donne de bien meilleurs résultats que l'utilisation du NDVI seul. En effet, les deux sont très importants pour détecter et localiser les arbres. Si nous détectons un objet qui a une certaine hauteur et qui est de la végétation, il y a de grandes chances que ce soit un arbre. Ces deux informations sont vraiment cruciales pour la détection des arbres.

Pour la tâche que nous voulons réaliser, le NDVI de par sa définition nous permet de ne conserver que la végétation présente dans l'image. Cela nous montre l'intérêt d'utiliser une source qui permet de mieux discriminer les arbres, même si cette source provient d'une transformation à partir des canaux de l'image optique. Pour autant, le CNN n'arrive pas à retrouver cette transformation lors de la phase d'entraînement à partir des images avec les canaux rouge, vert et proche infrarouge. Nous supposons que c'est à cause de la faible quantité de données que nous avons.

Il y a deux points très intéressants que l'on peut noter en regardant les résultats de la figure 4.6 : Tout d'abord sur l'image b) on peut voir que le faux positif le plus au centre de l'image est en réalité un arbre qui a été oublié par l'annotateur. Ensuite on peut aussi remarquer que dans les deux images, la majorité des faux négatifs, c'est-à-dire des arbres non détectés par le réseau, se trouvent en bordure de l'image. Ce phénomène vient du fait que lors de la phase d'apprentissage, les arbres "vus" par le réseau sont tous centrés avec quelques pixels de contexte autour. Différents travaux et notamment [Dundar et Garcia-Dorado, 2017] ont montré l'importance du contexte lors de l'entraînement des CNN. De plus ce phénomène vient aussi du fait que les arbres en bordure ne peuvent pas être centrés correctement dans les boîtes englobantes tout en ayant du contexte. Ainsi les arbres qui sont collés à la bordure dans les images de test ne seront pas détectés et localisés correctement. Une amélioration envisageable serait d'augmenter la base d'apprentissage en intégrant des imagettes dans lesquelles les arbres seraient "coupés" ou bien d'ajouter du padding.

Nous avons aussi calculé les corrélations entre les différentes sources. Ces expériences



(a) (b)

FIGURE 4.6 – Exemples de résultats obtenus, avec en vert les arbres correctement localisés, en bleu les faux négatifs et en jaune les faux positifs.

nous ont permis de déterminer si les différentes sources permettent d'améliorer les performances. On peut observer dans le tableau 4.4 que les deux corrélations sont d'environ 50%. Ces résultats montrent que parmi tous les arbres trouvés, environ 50% des arbres se trouvent dans les deux sources et que les 50% restants se trouvent dans la première ou la deuxième source. Ces résultats nous montrent la complémentarité qu'il existe entre les différentes sources.

De plus, la deuxième ligne du tableau montre que les 50% restants sont répartis dans les deux sources et nous montre ainsi l'utilité de combiner plusieurs sources.

Nous avons également calculé la corrélation des faux positifs entre les différentes sources et nous avons remarqué que cette corrélation ne dépasse jamais 10% quelles que soient les sources étudiées. Ainsi, la combinaison des sources réduit bien le nombre de faux positifs.

4.6 Conclusions

Dans ce chapitre, nous avons comparé deux méthodes permettant de combiner des données multi-sources (des images multispectrales composées des canaux rouge, vert et proche infrarouge et des MNS). Nous avons mis en compétition une approche "naïve" appelée ***Early Fusion*** qui consiste à mettre les différentes informations comme les composantes d'une image, à l'approche ***Late Fusion*** qui utilise un sous-réseau pour chaque source. Nous avons pu montrer avec nos expérimentations que l'approche naïve permettait d'obtenir les meilleurs performances en terme de F-mesure, de rappel et de précision.

De plus, au cours de nos expérimentations, nous avons pu évaluer l'impact que pouvait avoir les données sur les performances du CNN. Nous avons utilisé le NDVI au lieu d'utiliser les données avec les canaux Rouge, Vert et Proche Infrarouge. Nous avons mon-

tré que l'utilisation du NDVI permet d'obtenir les meilleures performances et met ainsi en évidence l'importance des données, de leurs pré-traitements ou même de leurs transformations pour apprendre un modèle avec un CNN. Nous avons pu constater que le réseau n'était pas capable de reproduire cette transformation lors de son apprentissage, cependant ce constat serait à valider avec une base d'apprentissage plus importante. En effet, nous pensons que la faible quantité de données que nous avons pour notre apprentissage ne permet pas au réseau de recréer cette transformation ou de l'améliorer.

Voici une procédure que l'on pourrait établir à partir des expériences et des observations que l'on a pu faire dans ce chapitre :

1. tester chaque source indépendamment,
2. calculer la corrélation entre toutes les paires de sources,
3. combiner les sources qui ont la corrélation la plus faible dans une approche EF ou LF.

Chapitre 5

Données incomplètes et réseau de neurones convolutionnels

Sommaire

5.1 Introduction

Dans le chapitre précédent, nous avons vu comment prendre en compte les données multi-sources. Mais que se passe-t-il si des sources sont manquantes ? En effet, jusque là nous avons toujours considéré que nos données étaient complètes, c'est-à-dire que nous avions accès à toutes les sources sur l'ensemble des données que nous utilisions. Nous avons pour ce chapitre réfléchi au problème de l'intégration et de l'utilisation des données incomplètes dans un réseau de neurones convolutionnels.

Les progrès concernant la technologie des capteurs ont permis de considérablement réduire le coût de production. Cette réduction de prix a permis d'utiliser plusieurs sources pour décrire un objet, et ainsi le décrire avec des perspectives différentes. Comme nous l'avons vu dans le chapitre 2, de nombreux travaux ont été réalisés afin de trouver la meilleure façon de combiner ces informations [Waltz et Llinas, 1990; Chong *et al.*, 2000; Liao *et al.*, 2013; Wang *et al.*, 2013] pour améliorer les performances sur différentes tâches et problématiques. La combinaison des informations a permis d'améliorer les performances sur les tâches de détection d'objets [Kanaev *et al.*, 2011], classification [Lim et Suter, 2009; Khodadadzadeh *et al.*, 2015] et de la segmentation sémantique [Elseberg *et al.*, 2011; Audebert *et al.*, 2016; Volpi et Tuia, 2017; Audebert *et al.*, 2018]. Nous avons d'ailleurs montré dans le chapitre 4 l'intérêt et l'amélioration des performances qui sont apportés par la combinaison de plusieurs sources.

Comme nous l'avons décrit dans le chapitre 2, lorsqu'il nous manque une source sur une ou plusieurs données et que nous souhaitons effectuer un apprentissage sur l'ensemble des données, il existe un certain nombre de solutions, dont (i) supprimer la source incomplète dans toutes les images (ii) supprimer les données qui n'ont pas toutes les sources (iii) générer des sources manquantes sur les données incomplètes [Shen *et al.*, 2015; Tran *et al.*, 2017]. Les deux premières solutions supprimeront beaucoup d'informations et cela peut engendrer une baisse des performances. Concernant la génération des sources manquantes, l'ajout de données est artificiel et donc pas totalement satisfaisant. D'autre part, si il n'y a que très peu d'exemples, les images générées ne seront pas significatives.

Contrairement aux chapitre 3 et 4, nous avons fait le choix dans ce chapitre de travailler sur la tâche de segmentation sémantique. Cela signifie que nous n'avons pas un label pour une imagette, mais que nous avons un label par pixel (voir la section 2.4 du chapitre 2). Durant l'entraînement, notre réseau doit donc apprendre à prédire le label de chaque pixel, et c'est aussi sur la prédiction des labels des pixels que nous évaluons notre réseau. Le fait de nous concentrer sur cette tâche nous a permis d'utiliser une base de données multimodales annotées plus importante que celle utilisée dans les chapitres précédents. La tâche de segmentation sémantique est très étudiée dans le domaine de la télédétection, notamment pour réaliser des analyses sur l'occupation du sol. L'utilisation d'un CNN est devenue courante pour ce type d'analyses [Volpi et Tuia, 2017; Maggiori *et al.*, 2017; Audebert *et al.*, 2018], cependant seul le cas où toutes les sources sont dispo-

nibles est considéré.

Dans ce chapitre, nous avons choisi de traiter, sur la tâche de segmentation sémantique, deux scénarios différents :

— Dans le premier scénario nous ciblons les images avec les canaux rouge et vert (RV), c'est-à-dire que lors de la phase de test, nous réaliserons nos tests sur des images RV. Dans la base d'apprentissage, toutes les images ont les canaux RV et nous supposons que seulement quelques unes ont aussi accès à la source proche infrarouge (Pir). Nous voulons que lorsqu'une donnée intègre la source Pir, cette dernière soit utilisée lors de l'apprentissage, et nous appelons ce scénario **l'enrichissement**. L'idée est d'enrichir notre base de connaissances en ajoutant une source lorsqu'elle est disponible, dans notre cas, en ajoutant le proche infrarouge.

— Dans le deuxième scénario, nous réaliserons nos tests sur des images PirRV. Dans ce scénario que nous appelons **le manque**, nous considérons lors de l'apprentissage comme dans le premier scénario que nous avons tout le temps accès à la source RV mais nous supposons que quelques fois il manque des données avec la source Pir. Lorsqu'une donnée n'a pas accès à la source Pir, nous donnons quand même l'image avec la source RV seule au réseau durant l'apprentissage. Nous voulons quand même utiliser les données en l'absence d'une source.

Le reste de ce chapitre est organisé comme suit : dans la section 5.2 nous présentons l'architecture que nous proposons. Dans la section 5.3 nous présentons le protocole expérimental utilisé et les résultats obtenus, et finalement nous concluons section 5.4.

5.2 Présentation de l'architecture à entrées multiples

Pour notre contribution, nous nous sommes inspirés des concepts presentés dans [Hou *et al.*, 2017] et [Kokkinos, 2017]. Les auteurs de [Hou *et al.*, 2017] utilisent deux sous-réseaux permettant d'extraire des caractéristiques (voir figure 5.1). Ces deux sous-réseaux sont connectés à une partie commune qui réalise la classification. L'idée de cette architecture est de coordonner les deux sous-réseaux afin qu'ils apprennent des caractéristiques complémentaires à partir des images qui leurs sont données. Leur but est d'extraire une meilleure représentation de l'image pour améliorer les performances. DualNet étant composé de 3 sorties, on peut voir cette approche comme un ensemble de classifieurs.

Dans [Kokkinos, 2017], l'auteur propose un seul CNN, appelé UberNet permettant de réaliser plusieurs tâches comme la détection, la segmentation sémantique ou bien l'extraction d'une carte de saillance, en utilisant un tronc commun, et plusieurs sorties qui réalisent des tâches différentes. Lors de l'apprentissage, si la vérité terrain d'une donnée n'est pas disponible pour une tâche, alors la sortie qui réalise cette tâche ne sera pas utilisée. La solution proposée pour pallier ce problème consiste à réaliser la rétropropagation de manière asynchrone. Cela signifie que ce n'est que lorsqu'une tâche a vu suffisamment d'exemples que l'on va appliquer la rétropropagation. L'idée est de rassembler dans

un espace latent commun tous les objets de la même classe quelle que soit la tâche.

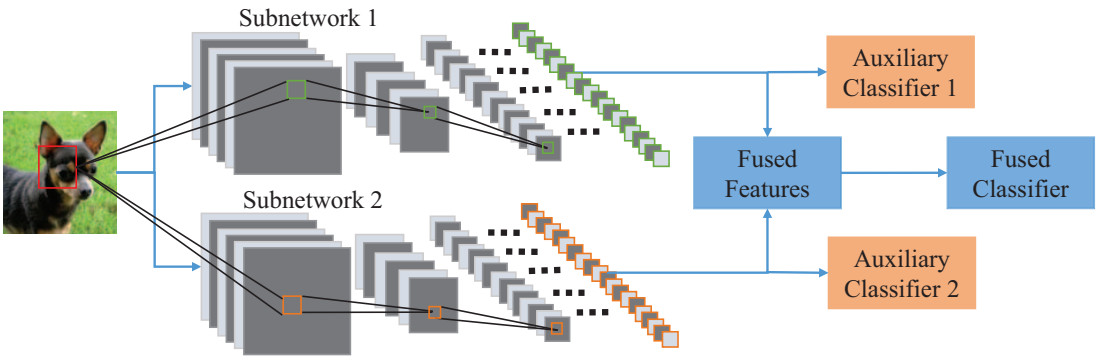


FIGURE 5.1 – Présentation de l'architecture DualNet. L'image donnée en entrée du réseau va être transmise à 2 sous-réseaux. Les *feature maps* des 2 sous-réseaux vont par la suite être données à des classifieurs qui sont indépendants à chaque sous-réseau, "Auxiliary Classifier 1" et "Auxiliary Classifier 2", mais les *feature maps* vont aussi être fusionnées et être données à un classifieur, "Fused Classifier". Source : [Hou *et al.*, 2017].

Nous avons appliqué ce concept sur l'architecture DeconvNet [Noh *et al.*, 2015] sur laquelle nous avons enlevé les couches *fully connected* entre l'encodeur et de décodeur (voir figure 5.2). Cette architecture est composée pour la partie encodeur de 13 couches de convolution, qui sont suivies d'une *batch normalization* (BN) et de la fonction d'activation ReLU et de 5 *max pooling*. Le décodeur est lui aussi composé de 13 couches de convolution qui sont suivies d'une *batch normalization* (BN) et de la fonction d'activation ReLU et de 5 couches de déconvolution.

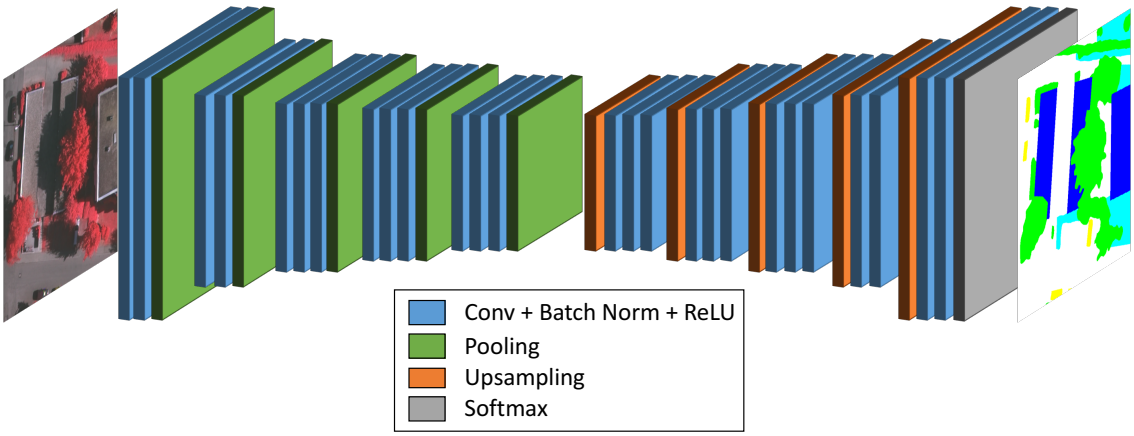


FIGURE 5.2 – Présentation de l'architecture DeconvNet. L'encodeur de ce réseau est composé de 13 couches de convolution (en bleu), qui sont suivies d'une BN et de la fonction d'activation ReLU et de 5 *pooling* (en vert). Le décodeur est quant à lui composé de 13 couches de convolution suivies d'une BN et de la fonction d'activation ReLU, de 5 couches de déconvolution et enfin d'un *softmax*.

Dans notre cas, les images données en entrée n'ont pas toujours le même nombre de canaux, c'est pourquoi nous avons adapté le concept d'utiliser plusieurs sous-réseaux à

notre problématique sur DeconvNet. L'architecture du CNN que nous proposons est présentée figure 5.3. Dans la suite du manuscrit, nous appelerons cette architecture VI-Net pour *Variable number of modalities as Input - Network*. Comme on peut le constater, nous avons repris l'architecture présentée figure 5.2, sauf qu'elle est composée de deux encodeurs (sur la partie gauche du schéma), et d'un commutateur (au milieu du schéma) qui connecte les deux encodeurs au décodeur. Chaque encodeur prend en entrée une source ou bien une combinaison de sources. Dans le schéma, l'encodeur en haut à gauche prend en entrée des images optiques avec les canaux RV, et l'encodeur en bas à gauche prend en entrée des images optiques avec les canaux RV combinés avec du Pir. Le commutateur nous permet de réaliser un apprentissage itératif. Cela signifie que l'on utilise nos encodeurs de manière alternée (on ne les utilise jamais en même temps), une fois on propage une image d'un encodeur jusqu'à la sortie du décodeur, puis on rétropropage sur le décodeur et l'encodeur utilisé, et à l'itération suivante on utilise l'autre encodeur. Le décodeur, quant à lui, est solicité à toutes les itérations.

L'idée de notre approche, est similaire à celle de [Kokkinos, 2017], par le biais de notre apprentissage itératif des encodeurs, nous cherchons à contraindre l'espace latent de représentation de nos *feature maps*. Quelle que soit la source utilisée, nous voulons que les objets de la même classe soient proches dans l'espace latent. Nous avons en quelque sorte inversé le concept de UberNet. Au lieu d'avoir un réseau à plusieurs sorties, nous avons un réseau avec plusieurs entrées qui partagent le même espace latent.

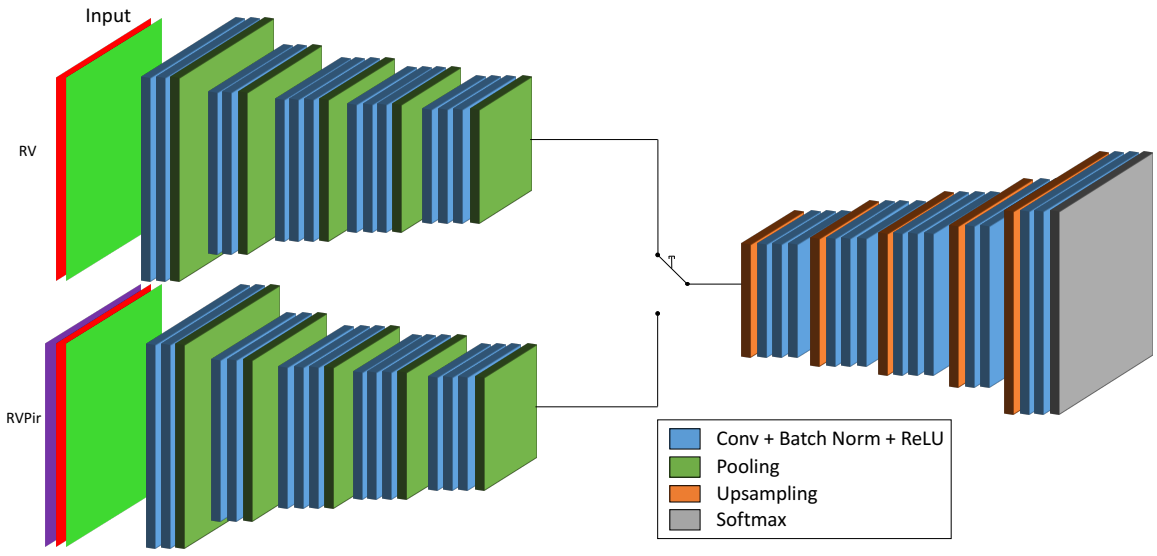


FIGURE 5.3 – Présentation de l'architecture proposée. La partie gauche du schéma représente les encodeurs qui sont composés de 13 couches de convolution et de 5 *max pooling*. L'encodeur du haut prend en entrée les images optiques RV, et l'encodeur du bas prend en entrée les images optiques RV + Pir. Au milieu du schéma, nous avons le commutateur nous permettant de faire notre apprentissage alterné. Enfin, à droite nous avons le décodeur composé de 13 couches de convolution et de 5 couches de déconvolution.

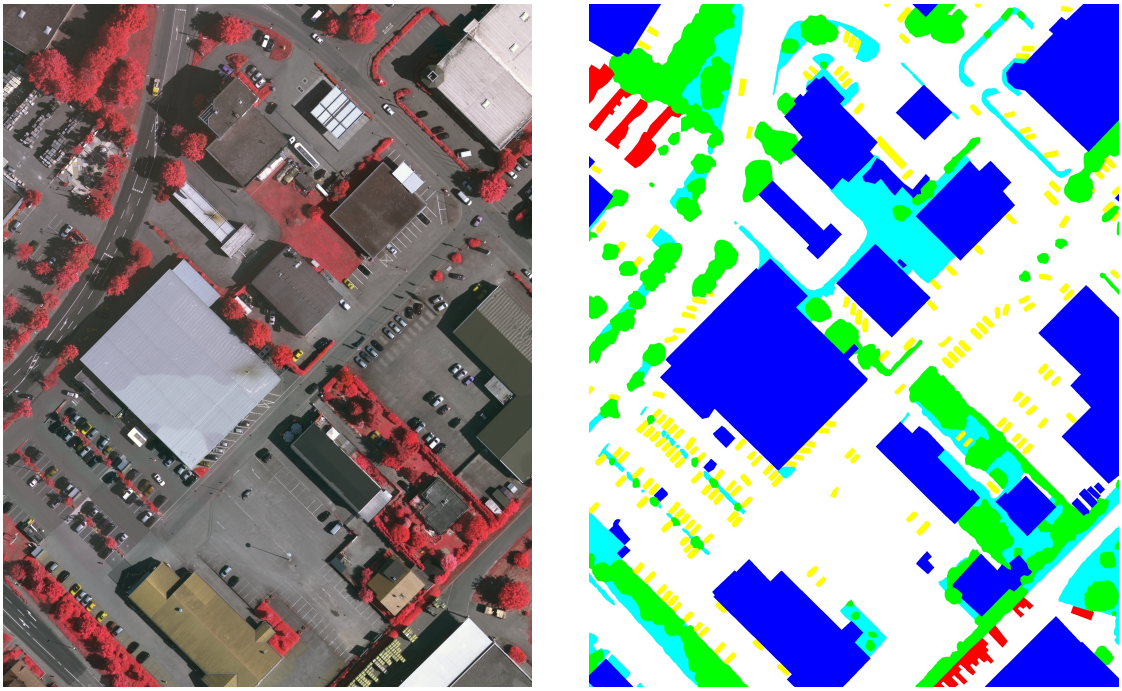
5.3 Évaluation expérimentale

5.3.1 Protocole expérimental

Pour réaliser nos expérimentations, nous avons divisé la base de données Vaihingen en deux parties. La première correspond à la base d'apprentissage qui nous permet d'entraîner nos modèles. Cette partie contient 14 images (voir figure 5.4). La seconde est composée de 2 images sur lesquelles nous testons nos modèles. Chaque image de la base d'entraînement possède environ 2500×2000 pixels. Les images que nous donnons aux entrées du réseau lors de la phase d'apprentissage sont des imagettes de 320×320 pixels qui sont extraits de façon aléatoire dans les images d'entraînement.

Notre base de test est composée de 2 images. Nous avons choisi de rogner la partie centrale des images de test à une taille de 1388×1388 pixels afin de s'assurer que quelles que soient les images utilisées lors de la phase de test, les résultats puissent être comparables.

Nous rappelons que contrairement aux chapitre 3 et 4, ici nous ne faisons pas un apprentissage objet mais pixel. Cela signifie que nous n'avons pas un label pour une imagette, mais que nous avons un label par pixel. Durant l'entraînement, notre réseau doit donc apprendre à prédire le label de chaque pixel, et durant la phase de test c'est aussi au niveau de la prédiction du label des pixels que nous évaluons notre réseau.



(a) (b)

FIGURE 5.4 – Exemples d'images données au CNN. Nous avons à gauche une image PirRV et à droite la vérité terrain associée. Dans la vérité terrain il y a les classes : *building* en bleu, *car* en jaune, *clutter / background* en rouge, *low vegetation* en turquoise, *tree* en vert et *impervious surfaces* en blanc.

Pour réaliser nos expérimentations sur les applications de l'enrichissement et du manque présentées plus haut, nous avons supprimé la source Pir d'un certain nombre d'images, et nous avons fait varier ce nombre entre 1 et 13 pour tester les cas de l'absence à la présence totale de la source Pir dans les 14 images. Lorsque le nombre d'images Pir est de 0 ou de 14, nous n'utilisons plus qu'un seul encodeur ce qui revient à utiliser DeconvNet. Un exemple des images utilisées lors de la phase d'apprentissage est donné figure 5.5. L'image de gauche de la figure est l'image PirRV et à droite l'image correspondante lorsque l'on supprime le Pir. Afin de pallier le petit nombre d'images de notre base, nous avons divisé notre base en 8 groupes de 2 images et nous avons réalisé l'apprentissage sur 7 groupes et testé sur le dernier groupe. Nous avons répété cette opération 5 fois.

Pour évaluer les résultats, nous avons utilisé le rappel pondéré, noté $\overline{\mathrm{R}}$, la précision pondérée, notée $\overline{\mathrm{P}}$, ainsi que la F-Mesure pondérée notée $\overline{\mathrm{F}}$ afin de prendre en compte les différences en nombres de pixels de chaque classe :

$$\overline{\mathrm{R}} = \frac{1}{\sum_{l \in \mathrm{L}} |y_l|} \sum_{l \in \mathrm{L}} |y_l| \left(\frac{|\hat{y}_l \cap y_l|}{|y_l|} \right), \tag{5.1}$$

$$\overline{\mathrm{P}} = \frac{1}{\sum_{l \in \mathrm{L}} |y_l|} \sum_{l \in \mathrm{L}} |y_l| \left(\frac{|\hat{y}_l \cap y_l|}{|\hat{y}_l|} \right), \tag{5.2}$$

$$\overline{\mathrm{F}} = \frac{2\overline{\mathrm{RP}}}{\overline{\mathrm{P}} + \overline{\mathrm{R}}}, \tag{5.3}$$

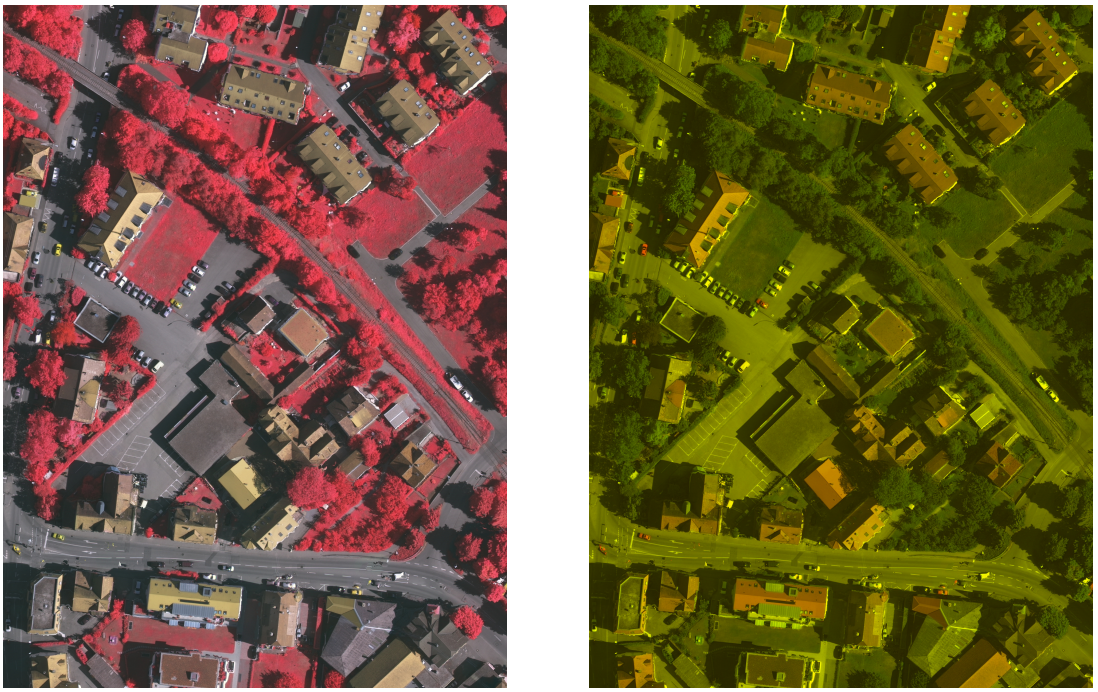
avec L l'ensemble des labels, y_l l'ensemble des pixels appartenant à la classe l, et $\hat{y}_l$ l'ensemble des pixels prédits par le réseau appartenant à la classe l.

5.3.2 Résultats

5.3.2.1 Résultats de référence

Afin de contextualiser nos résultats, nous avons réalisé nos premières expérimentations sur l'architecture DeconvNet [Noh *et al.*, 2015] (présentée figure 5.2). Nous avons utilisé cette architecture lorsque nous utilisons uniquement des images RV ou bien uniquement des images PirRV.

Le tableau 5.1 présente les résultats obtenus lorsque nous utilisons l'architecture DeconvNet, c'est-à-dire lorsque l'on utilise un nombre fixe de sources lors de la phase d'apprentissage. Ce tableau présente les résultats lorsque nous effectuons la phase d'apprentissage et la phase de test sur les images RV pour la deuxième colonne et sur les images PirRV pour la troisième colonne. Comme on peut le constater, les résultats obtenus sont meilleurs lorsque l'on utilise toutes les sources, c'est-à-dire les données PirRV. Comme on peut le constater dans le tableau 5.1, la suppression de l'information Pir fait diminuer la précision de 4% et le rappel de 7%. Ces résultats nous permettent de confirmer l'importance de l'information Pir et souligne son pouvoir discriminatif pour notre tâche. On considère pour le reste des nos expériences qu'il s'agit des bornes minimale et maximale



(a) (b)

FIGURE 5.5 – Exemples d'images données au CNN, avec à gauche une image PirRV et à droite la même image avec seulement les canaux RV. Comme on peut le constater sur ces deux images, on distingue plus facilement sur l'image PirRV que sur l'image RV.

puisque les résultats avec les données RV correspondent au cas où nous avons le moins d'informations et sur les données PirRV au cas où nous avons accès à toutes les informations.

La deuxième est l'architecture que nous proposons que nous appelons VI-Net et qui est présentée figure 5.3. Nous avons utilisé cette architecture lorsque nous avons réalisé l'apprentissage sur des images RV et des images PirRV, c'est-à-dire sur les scénarios de l'enrichissement et du manque.

5.3.2.2 Scénario de l'enrichissement

La figure 5.6 représente les expériences que nous avons menées sur le scénario de l'enrichissement. Dans ce scénario, nous effectuons les tests sur les images RV et nous regardons l'infuence de l'introduction d'une source supplémentaire en faisant varier dans la base d'apprentissage le nombre d'image PirRV de 1 à 13. Sur la figure 5.6 nous avons ajouté les F-mesures que nous avons présentées dans le tableau 5.1 avec les données RV et PirRV. Ces résultats sont représentés en bleu et en rouge dans le schéma et représentent la borne minimale et la borne maximale. Les résultats que nous avons obtenus sur le scénario de l'enrichissement sont représentés en vert sur la figure. En abscisse nous avons mis le nombre d'images PirRV utilisées durant la phase d'apprentissage et en ordonnée la F-mesure obtenue sur la base de test. Comme on peut le constater sur la figure 5.6, ajouter la source Pir sur une sous-partie de la base d'apprentissage permet d'améliorer les

Entraînement	RV	PirRV
F-Mesure	68,40%	**74,56%**
Rappel	66,73%	**73,87%**
Précision	71,35%	**75,38%**

TABLEAU 5.1 – Résultats obtenus sur la base de données Vaihingen sur une tâche de segmentation sémantique. Ce tableau présente les résultats obtenus en utilisant les 14 images pour la phase d'entraînement et les 2 images de test avec les données PirRV et les données RV. Dans ces expériences nous avons utilisé un nombre de sources fixe, nous avons donc utilisé l'architecture DeconvNet.

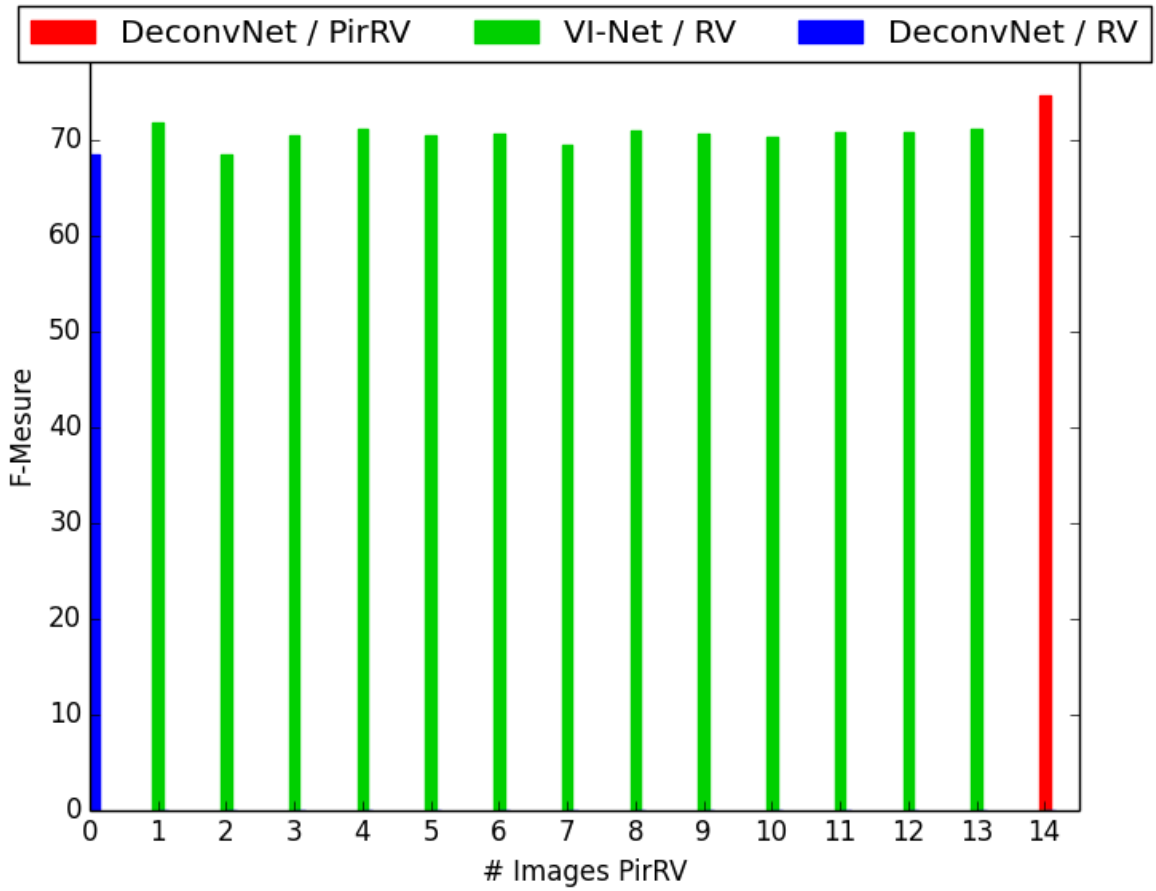


FIGURE 5.6 – Représentation des résultats obtenus sur le scénario de l'enrichissement. Le test est réalisé sur les images optiques RV et l'entraînement est réalisé avec VI-Net sur les images optiques RV et PirRV lorsque le Pir est disponible (en vert). Nous avons représenté en bleu la borne minimale et en rouge la borne maximale. La borne minimale correspond à la F-mesure obtenue en utilisant uniquement les images RV (14 images RV), et en rouge la F-mesure obtenue en utilisant uniquement les données PirRV (14 images PirRV). En abscisse nous avons le nombre d'images PirRV utilisées lors de la phase d'apprentissage et en ordonnée nous avons la F-mesure obtenue sur la base de test.

résultats lorsque l'on teste sur des images RV. En effet, à l'exception du cas où nous avons

utilisé deux images PirRV où nous n'avons ni gain ni perte, on peut voir que l'on obtient toujours une F-mesure supérieure à celle obtenue en utilisant uniquement des images RV. Ces résultats montrent que l'utilisation du Pir lors de la phase d'apprentissage, permet d'améliorer les résultats dans le scénario de l'enrichissement. En effet, on peut noter une amélioration des résultats puisque nous avons en moyenne un gain de 2%, ce qui nous permet de passer d'une F-mesure de 68,4% à 70,49% en utilisant des images PirRV durant l'apprentissage.

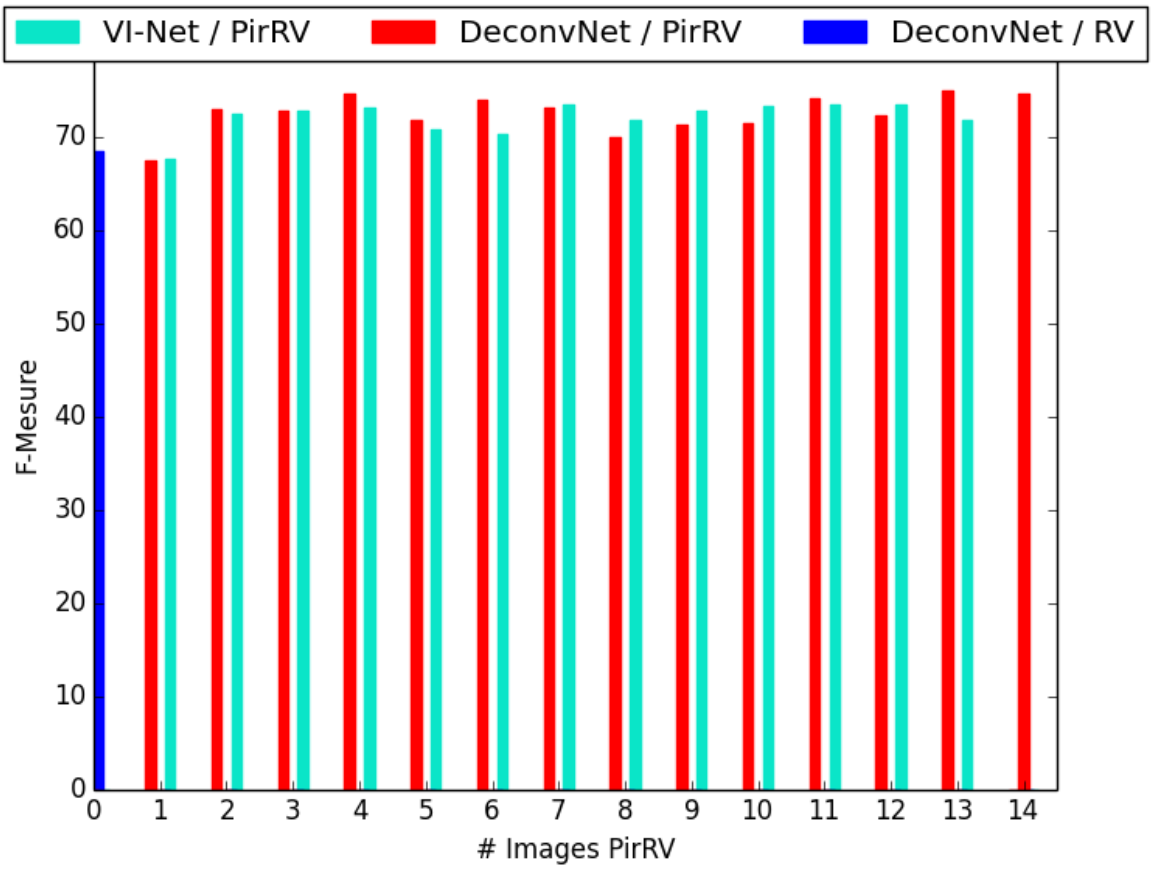


FIGURE 5.7 – Représentation des résultats obtenus sur le scénario du manque. Le test est réalisé sur le PirRV et l'entraînement est réalisé avec VI-Net sur les images optiques RV et PirRV lorsque le Pir est disponible (en turquoise). En rouge, nous avons représenté les résultats obtenus avec DeconvNet entraîné uniquement sur les image PirRV disponibles. Nous avons représenté en bleu la borne minimale, c'est-à-dire la F-mesure obtenue en utilisant uniquement les images RV. Les résultats sur le scénario du manque sont représentés en turquoise dans la figure. En abscisse nous avons le nombre d'images PirRV utilisées lors de la phase d'apprentissage et en ordonnée nous avons la F-mesure obtenue sur la base de test.

5.3.2.3 Scénario du manque

La figure 5.7 représente les expériences que nous avons menées sur le scénario du manque. Comme pour l'expérience précédente, nous avons 14 images RV et nous avons fait varier le nombre d'images avec la source Pir. Dans ce scénario, nous avons réalisé tous

nos tests sur des images PirRV. Comme pour la figure 5.6, nous avons représenté en bleu la borne minimale et en rouge la borne maximale. Dans le cas du scénario du manque, pour valider nos résultats nous avons aussi réalisé des tests en utilisant uniquement les images PirRV (entre 1 et 13 images) pendant l'apprentissage avec l'architecture DeconvNet. Cela signifie que si une image n'a pas accès à toutes les sources, elle ne sera pas utilisée lors de la phase d'apprentissage. Ces expérimentations nous ont permis de déterminer si le fait d'utiliser des images qui n'ont pas la source Pir peut aider durant l'apprentissage et ainsi permettre d'obtenir de meilleures résultats.

Comme on peut le constater sur la figure 5.7, notre architecture permet dans certains cas d'améliorer les performances. En effet, on peut voir que lorsque nous avons accès au Pir sur par exemple 8 images, utiliser en plus des images RV durant l'apprentissage permet d'améliorer les résultats en passant d'une F-mesure de 69.92% à 71,83%.

Cependant, on peut noter que dans d'autres cas, l'ajout des images RV durant l'apprentissage empêche le réseau d'apprendre correctement et fait baisser les performances. Par exemple lorsque nous avons utilisé 6 et 13 images PirRV lors de la phase d'apprentissage, les résultats en utilisant uniquement les images PirRV durant l'apprentissage sont bien plus élevés que lorsque l'on utilise les images RV en plus. Lorsque l'apprentissage est réalisé sur seulement 6 images PirRV la F-mesure est de 73,86% contre 70,17% lorsque l'on utilise les images RV, et avec 13 images PirRV on passe d'une F-mesure de 74.89% à 71,79%.

Pour ce scénario les résultats sont donc plus contrastés que pour le scénario précédent. Toutefois, lorsque l'on fait la moyenne des résultats, on obtient une F-mesure de 72.34% lorsque l'on entraîne notre réseau avec des images PirRV+RV contre une F-mesure de 72.06% avec DeconvNet uniquement entraîné avec des images PirRV.

5.3.2.4 Résumé des expérimentations

La figure 5.8 présente un récapitulatif des résultats que nous avons obtenus sur ces scénarios et lorsque nous utilisons uniquement des images RV ou bien uniquement des images PirRV. En bleu, il s'agit du cas où nous n'avons utilisé que des images RV et en vert le scénario de l'enrichissement avec le nombre d'images PirRV que nous avons utilisé durant l'apprentissage. En rouge, nous n'utilisons que les images PirRV durant l'apprentissage et en bleu clair nous avons le cas du manque où nous utilisons des images PirRV et RV durant l'apprentissage.

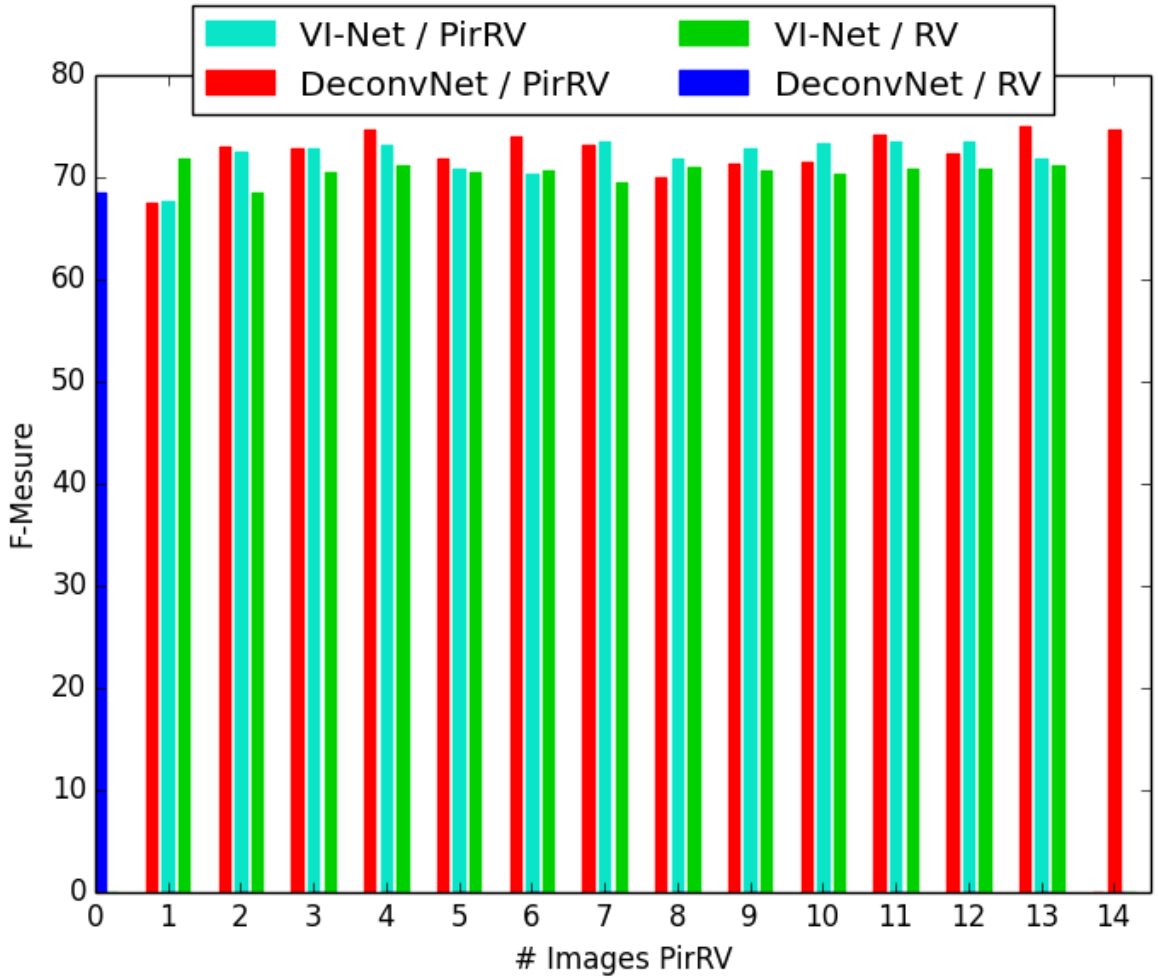


FIGURE 5.8 – Représentation des résultats obtenus sur le scénario du manque. Nous avons représenté en bleu la borne minimale et en rouge la borne maximale. La borne minimale correspond à la F-mesure obtenue en utilisant uniquement les images RV (la borne minimale), et en rouge la F-mesure obtenue en utilisant uniquement les données PirRV. La borne minimale et la borne maximale ont été obtenues sur l'architecture DeconvNet. Les résultats sur le scénario du manque sont représentés en vert dans la figure. En abscisse nous avons le nombre d'images PirRV utilisées lors de la phase d'apprentissage et en ordonnée nous avons la F-mesure obtenue sur la base de test. Dans le cas du scénario du manque, les tests sont réalisés sur les images PirRV.

5.4 Conclusions

Dans ce chapitre, nous avons proposé une nouvelle architecture que nous avons appelé VI-Net. Cette nouvelle architecture permet de réaliser un apprentissage en utilisant un seul réseau de neurones convolutionnels mêmes lorsque nous avons des données incomplètes, c'est-à-dire quand nous n'avons pas accès à toutes les sources de tous les éléments de la base d'apprentissage. Notre architecture est basée sur le réseau DeconvNet [Noh *et al.*, 2015] mais peut facilement être adaptée à d'autres architectures.

Afin de valider notre approche et notre architecture, nous avons réalisé des expéri-

mentations sur deux scénarios, celui de l'enrichissement et celui du manque. Dans le scénario de l'enrichissement, nous ciblons les données avec la source RV. Cela signifie que notre objectif est d'améliorer les résultats sur les images RV de la base de test, et pour ce faire nous utilisons des images PirRV durant la phase d'apprentissage pour enrichir notre modèle. Durant cette expérience, nous avons pu constater que dans le cas de l'enrichissement notre approche permet d'améliorer la F-mesure de plus de 2%.

Dans les cas du scénario du manque, nous ciblons les images PirRV pendant la phase de test. L'idée derrière ce scénario est d'utiliser l'image RV même lorsque nous n'avons pas accès à la source Pir. Comme nous l'avons vu précédemment, les résultats de ce scénario sont plus contrastés.

Nous avons montré dans ce chapitre par le biais de nos expérimentations que notre architecture permet d'améliorer les performances dans le scénario de l'enrichissement. Ces résultats montrent que l'utilisation de toutes les données, même lorsqu'elles sont incomplètes est bénéfique lors de l'entraînement d'un réseau de neurones convolutionnels. Ces résultats sont prometteurs et demanderaient à être approfondis et validés par d'autres expériences, et notamment par l'utilisation de plus de sources et plus de données. Nous pourrions par exemple appliquer VI-Net sur les images de la base de données ImageNet en donnant en entrée d'un sous-réseau l'image RV et l'image B dans un deuxième sous-réseau. Cette base de données étant composée de plus d'un million d'images, cela nous permettrait de valider nos résultats.

Chapitre 6

Conclusions et perspectives

6.1 Bilan

Dans cette thèse, nous nous sommes focalisés sur la localisation dans des images aériennes multi-sources (optique, proche infrarouge et MNS) d'un objet urbain particulièrement variable en termes d'apparence : l'arbre urbain.

Nous avons commencé par réaliser une étude afin de déterminer si les CNNs pouvaient être une solution envisageable pour la tâche de localisation d'arbres urbains. Pour cela, nous avons mis en place une procédure d'analyse locale multi-échelles par fenêtre glissante dans l'image. Pour l'analyse locale, nous avons comparé des approches classiques d'apprentissage automatique en deux étapes (descripteur fondé sur des cascades de HOG combiné avec un classifieur SVM et Random Forest) avec deux architectures de CNN, ALexNet et GoogleNet. Nous avons aussi étudié l'impact de la méthode de fusion des résultats de l'analyse locale. Cette étude nous a permis de confirmer que l'approche par *deep learning* est une approche viable pour la tâche de localisation d'arbres urbains.

À partir de ce constat, nous avons cherché à améliorer les résultats obtenus par les CNNs lors de notre étude en travaillant sur la manière de combiner les différentes sources des données. Pour cela nous avons comparé l'approche ***Early Fusion*** qui consiste à simplement concaténer les sources à une approche ***Late Fusion*** où les sources sont traitées en parallèle jusqu'à la classification. Nous avons fait une analyse sur l'utilisation des différentes sources et nous avons pu défendre la thèse selon laquelle l'utilisation de données multi-sources a un intérêt sur la tâche de localisation d'arbres urbains. En particulier, nous avons pu montrer la complémentarité qu'il peut exister entre les différentes sources des données. Nous montrons que l'approche ***Early Fusion*** en utilisant les sources NDVI (une combinaison des sources RV et infrarouge, très utilisée en télédétection de végétation) et MNS permet d'obtenir les meilleurs résultats mais il faudrait poursuivre les expériences avec une base de données plus conséquente. En effet, nous pouvons penser que le CNN est capable de combiner automatiquement dans une approche ***Late Fusion*** les sources d'origine RV et infrarouge pour créer un équivalent du NDVI.

Nous avons alors choisi de nous intéresser au problème des données incomplètes. En effet, jusque-là nous avions considéré que nous avions accès à l'ensemble des sources sur toutes les données mais dans certains cas (par exemple dans le cas d'un capteur défaillant), des sources peuvent être manquantes, on peut donc avoir accès à l'image optique sans avoir le proche infrarouge. Pour cette contribution nous avons réfléchi à une stratégie nous permettant de traiter la phase d'apprentissage et de test sur la totalité des données (c'est-à-dire sans supprimer les données incomplètes) sans générer les sources manquantes. Nous avons proposé une nouvelle architecture permettant de prendre en entrée des données avec un nombre variable de sources et en réalisant un apprentissage alterné.

Nous avons montré durant nos expérimentations que lorsque nous sommes dans le scénario de l'enrichissement, c'est-à-dire lorsque nous ciblons la source où nous avons accès à toutes les données et nous utilisons la ou les sources incomplètes pour enrichir

notre base d'apprentissage, notre approche permettait d'améliorer les résultats.

6.2 Perspectives

Nous présentons dans cette section une liste de plusieurs améliorations des méthodes présentées dans la thèse, ainsi que des pistes de recherche qu'il serait intéressant d'explorer.

Dans le chapitre 5, nous nous sommes inspirés du concept de l'architecture DualNet en utilisant 2 sous-réseaux permettant de prendre une source, ou bien une combinaison de sources pour chaque entrée d'un sous-réseau. Nous pourrions étendre cette architecture en considérant que nous avons un réseau multi-entrées mais aussi multi-sorties. Nous aurions ainsi un décodeur pour chaque sous-réseau en plus du décodeur commun aux 2 sous-réseaux. L'architecture d'un tel réseau est illustrée figure 6.1. Si nous avons soit la source RV, soit la combinaison des sources Pir et RV, le réseau aura donc 2 entrées et 3 sorties, une sortie pour chaque sous-réseau et une sortie commune. Si nous avons plus de sources à notre disposition, il est possible d'augmenter le nombre de sous-réseaux et le nombre de sorties.

De plus, avec une telle architecture il est très simple de pondérer les fonctions de perte suivant les données en entrée et le scénario choisi durant l'apprentissage. Cette architecture nous permettrait de favoriser une source suivant le scénario choisi. Par exemple dans le scénario de l'enrichissement, nous pourrions réduire l'importance de la fonction de perte (en la divisant par 2 par exemple) lorsque les données en entrée sont la combinaison des sources Pir et RV. Cette stratégie permettrait de contraindre l'espace latent de représentation, c'est-à-dire l'espace dans lequel nos données sont représentées, tout en "forçant" le réseau à tendre vers une solution qui favorise la source qui aura le poids le plus important.

Nous avons réalisé des tests préliminaires en utilisant l'architecture présentée figure 6.1. Nous avons suivi le même protocole expérimental que dans le chapitre 5 et nous avons utilisé uniquement la prédiction du classifieur RV + RVPir. En ce qui concerne les pondérations des trois fonctions de perte, la fonction de perte du classifieur RV + RVPir a un poids de 1 et les deux autres classifieurs ont un poids de 0,3. Les premiers résultats montrent une amélioration de la F-mesure de plus de 1% sur les scénarios du manque et de l'enrichissement (voir chapitre 5).

Une autre stratégie pour améliorer les résultats présentés dans le chapitre 5, serait d'utiliser le même principe que les réseaux siamois [Bromley *et al.*, 1994; Chopra *et al.*, 2005]. Nous réaliserions un premier apprentissage qui forcerait le réseau à placer les objets de la classe dans le même espace latent en réduisant la distance entre les *feature maps* de la même classe quelques soient les sources données en entrée. Ce type d'approche a déjà été utilisé dans [He *et al.*, 2018], où les auteurs utilisent un réseau siamois pour faire de la mise en correspondance d'images. Une fois ce premier apprentissage réalisé, nous

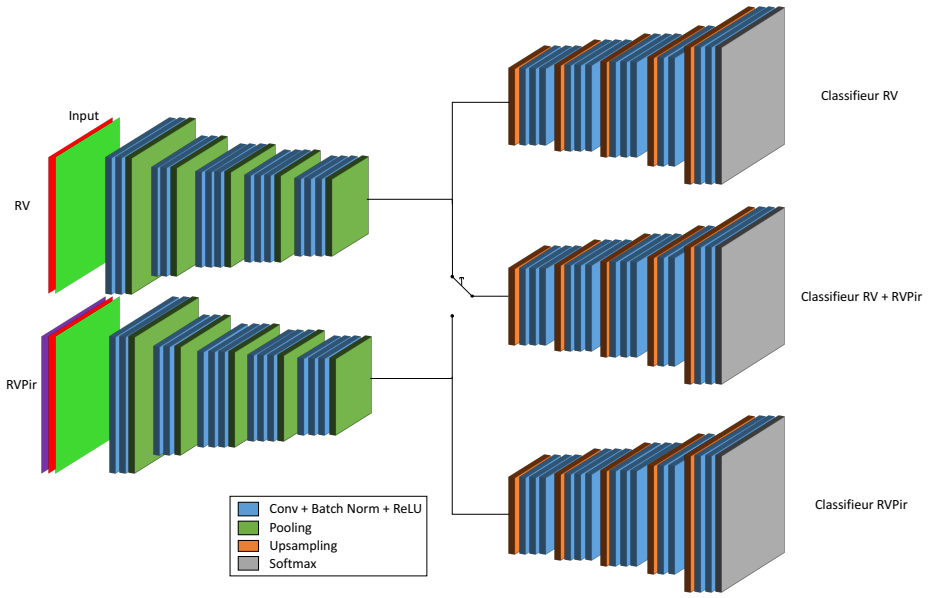


FIGURE 6.1 – Exemple d'architecture en adaptant DualNet sur le problème des données incomplètes. La partie gauche du schéma représente les encodeurs qui sont composés de 13 couches de convolution et de 5 *max pooling*. L'encodeur du haut prend en entrée les images optiques RV, et l'encodeur du bas prend en entrée les images optiques RV + Pir. Au milieu du schéma, nous avons le commutateur nous permettant de faire l'apprentissage alterné sur le décodeur commun prenant en entrée des données RV et RVPir. À droite nous avons les 3 décodeurs composés de 13 couches de convolution et de 5 couches de déconvolution. Le décodeur du haut n'est utilisé que lorsque ce sont des images optiques RV qui sont données en entrée, et celui du bas n'est utilisé que lorsque ce sont des données RVPir qui sont données en entrée. Enfin le décodeur du milieu est utilisé à chaque itération, quelles que soient les données utilisées.

ferions un deuxième apprentissage en suivant le même protocole expérimental que dans le chapitre 5.

Une autre piste de recherche serait d'adapter l'approche présentée dans [He *et al.*, 2017a] à un problème multi-sources. Cette approche permet de réaliser une tâche d'*instance segmentation*, c'est-à-dire de détecter et de délimiter chaque objet distinct apparaissant dans une image (voir figure 6.2). Les auteurs de cet article utilisent un réseau qu'ils ont appelé *Mask R-CNN* pour réaliser cette tâche. *Mask R-CNN* utilise les mêmes étapes que *Faster R-CNN* (voir section 2.3), c'est-à-dire la génération des régions d'intérêt, et ensuite la classification des régions d'intérêt et l'estimation des coordonnées des boîtes englobantes, mais en plus *Mask R-CNN* extrait un masque binaire sur chaque région d'intérêt. Des exemples de résultats obtenus avec *Mask R-CNN* sont présentés figure 6.2. Dans le cas de la localisation d'arbres urbains qui sont souvent partiellement occultés, cette approche nous permettrait d'avoir une segmentation plus fine des arbres, ce qui est plus fin que les boîtes englobantes que nous avons utilisées dans les chapitres 3 et 4.

FIGURE 6.2 – Exemples de résultats qui ont été obtenus en utilisant le réseau *Mask R-CNN*. Source : [He *et al.*, 2017a].

Publications

- Benjamin COMMANDRE, Driss EN-NEJJARY, Lionel PIBRE, Marc CHAUMONT, Gérard SUBSOL, Laurent DERUELLE, Mustapha DERRAS, Carole DELENNE, Nanée CHAHINIAN : Détection de regards de visite sur des images à très haute résolution spatiale par une méthode d'apprentissage. *Atelier Télédétection pour l'Etude des Milieux Urbains (TEMU)*, Strasbourg, France, mars 2018.

- Lionel PIBRE, Marc CHAUMONT, Gérard SUBSOL, Dino IENCO, and Mustapha DERRAS : How to deal with multi-source data for tree detection based on deep learning. *IEEE Global Conference on Signal anf Information Processing (GlobalSIP)*, Montréal, Canada, novembre 2017.

- Benjamin COMMANDRE, Driss EN-NEJJARY, Lionel PIBRE, Marc CHAUMONT, Carole DELENNE, Nanée CHAHINIAN : Manhole Cover Localization in Aerial Images with a Deep Learning Approach. *International Society for Photogrammetry and Remote Sensing Workshop (ISPRS)*, Hannover, Allemagne, juin 2017.

- Lionel PIBRE, Marc CHAUMONT, Gérard SUBSOL, Dino IENCO, and Mustapha DERRAS : Detection of Urban Trees in Multiple-Source Aerial Data (Optical, Infrared, DSM). *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) M.Sc./Ph.D. Forum*, Nouvelle-Orléans, USA, mars 2017.

- Lionel PIBRE, Marc CHAUMONT, Gérard SUBSOL, Dino IENCO, and Mustapha DERRAS : Détection d'arbres urbains à partir de données aériennes multi-sources (optique, infrarouge, MNS). *Atelier Télédétection pour l'Etude des Milieux Urbains (TEMU)*, Toulouse, France, janvier 2017.

- Lionel PIBRE, Marc CHAUMONT, Dino IENCO, et Jérôme PASQUET : Étude des réseaux de neurones sur la stéganalyse. *COmpression et REprésentation des Signaux Audiovisuels (CORESA)*, Nancy, France, mai 2016.

- Lionel PIBRE, Marc CHAUMONT, Dino IENCO, et Jérôme PASQUET : Deep learning is a good steganalysis tool when embedding key is reused for different images, even if there is a cover source-mismatch. *International Symposium on Electronic Imaging (EI)*, San Francisco, USA, février 2016.

Bibliographie

Mongi A. ABIDI et Rafael C. GONZALEZ : *Data Fusion in Robotics and Machine Intelligence*. Academic Press Professional, Inc., San Diego, CA, USA, 1992. 51

Ashutosh AGGARWAL, Karamjeet SINGH et Kamalpreet SINGH : Use of gradient technique for extracting features from handwritten gurmukhi characters and numerals. *Procedia Computer Science*, 46:1716 – 1723, 2015. 15, 16

David ALDAVERT, Ramon Lopez DE MANTARAS, Arnau RAMISA et Ricardo TOLEDO : Fast and robust object segmentation with the integral linear classifier. *In IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1046–1053, San Francisco, USA, juin 2010. 4

David ALDAVERT, Arnau RAMISA, Ricardo TOLEDO et Ramon Lopez DE MANTARAS : Efficient object pixel-level categorization using bag of features. *In International Symposium on Visual Computing*, pages 44–54, 2009. 27

Michael ALONZO, Bodo BOOKHAGEN et Dar A. ROBERTS : Urban tree species mapping using hyperspectral and LiDAR data fusion. *Remote Sensing of Environment*, 148:70–83, 2014. 76

Ashima ANAND : Brain tumor segmentation using watershed technique and self organizing maps. *Indian Journal of Science and Technology*, 10(44), 2017. 49

Pablo ARBELAEZ, Michael MAIRE, Charless FOWLKES et Jitendra MALIK : Contour detection and hierarchical image segmentation. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, 2011. 48

Nicolas AUDEBERT, Bertrand LE SAUX et Sébastien LEFÈVRE : Semantic segmentation of earth observation data using multimodal and multi-scale deep networks. *In Asian Conference on Computer Vision (ACCV)*, pages 180–196, Taipei, Taiwan, novembre 2016. 51, 87

Nicolas AUDEBERT, Bertrand LE SAUX et Sébastien LEFÈVRE : Beyond rgb : Very high resolution urban remote sensing with multimodal deep networks. *ISPRS Journal of Photogrammetry and Remote Sensing*, 140:20–32, 2018. 51, 87

Pedram AZAD, Tamim ASFOUR et Rüdiger DILLMANN : Combining harris interest points and the sift descriptor for fast scale-invariant object recognition. *In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4275–4280, St. Louis, USA, octobre 2009. 21

Cédric BACOUR, François-Marie BRÉON et Fabienne MAIGNAN : Normalization of the directional effects in noaa–avhrr reflectance measurements for an improved monitoring of vegetation cycles. *Remote Sensing of Environment*, 102(3-4):402–413, 2006. 75

Richard BARNES, Clarence LEHMAN et David MULLA : Priority-flood : An optimal depression-filling and watershed-labeling algorithm for digital elevation models. *Computers & Geosciences*, 62:117–127, 2014. 49

Mariana BELGIU et Lucian DRĂGUŢ : Random forest in remote sensing : A review of applications and future directions. *ISPRS Journal of Photogrammetry and Remote Sensing*, 114:24–31, 2016. 29

Asa BEN-HUR, David HORN, Hava SIEGELMANN et Vladimir VAPNIK : Support vector clustering. *Journal of Machine Learning Research*, 2(Dec):125–137, 2001. 25

Yoshua BENGIO, Aaron COURVILLE et Pascal VINCENT : Representation learning : A review and new perspectives. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013. 7

Luca BERTELLI, Tianli YU, Diem VU et Burak GOKTURK : Kernelized structural svm learning for supervised object segmentation. *In IEEE conference on Computer Vision and Pattern Recognition (CVPR), Denver, USA*, pages 2153–2160, juin 2011. 49

Serge BEUCHER : Use of watersheds in contour detection. *In Proceedings of the International Workshop on Image Processing*, 1979. 48

Bernhard BOSER, Isabelle GUYON et Vladimir VAPNIK : A training algorithm for optimal margin classifiers. *In Proceedings of the Workshop on Computational learning theory*, pages 144–152, 1992. 7, 24, 27

Leo BREIMAN : Bagging predictors. *Machine learning*, 24(2):123–140, 1996. 30

Leo BREIMAN : Random forests. *Machine Learning*, 45(1):5–32, 2001. 29

Alberto BROGGI, Pietro CERRI, Paolo MEDICI, Pier Paolo PORTA et Guido GHISIO : Real time road signs recognition. *In IEEE Intelligent Vehicles Symposium*, pages 981–986, 2007. 56, 59

Jane BROMLEY, Isabelle GUYON, Yann LECUN, Eduard SÄCKINGER et Roopak SHAH : Signature verification using a "siamese" time delay neural network. *In Advances in neural information processing systems*, pages 737–744, 1994. 101

Matthew BROWN et David G. LOWE : Invariant features from interest point groups. *In British Machine Vision Conference (BMVC)*, volume 4, Cardiff, UK, septembre 2002. 19, 20

Chih-Chung CHANG et Chih-Jen LIN : Libsvm : a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011. 26, 29

Vinod Kumar CHAUHAN, Kalpana DAHIYA et Anuj SHARMA : Problem formulations and solvers in linear svm : a review. *Artificial Intelligence Review*, pages 1–53, 2018. vi, 27, 28

Liang-Chieh CHEN, George PAPANDREOU, Iasonas KOKKINOS, Kevin MURPHY et Alan L YUILLE : Deeplab : Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2018. vi, vii, 41, 50

Yunpeng CHEN, Jianan LI, Huaxin XIAO, Xiaojie JIN, Shuicheng YAN et Jiashi FENG : Dual path networks. *In Advances in Neural Information Processing Systems (NIPS)*, Long Beach, USA. 45

KyungHyun CHO : Simple sparsification improves sparse denoising autoencoders in denoising highly corrupted images. *In International Conference on Machine Learning (ICML)*, pages 432–440, Atlanta, USA, juin 2013. 53

Chee-Yee CHONG, Shozo MORI, William H BARKER et Kuo-Chu CHANG : Architectures and algorithms for track association and fusion. *IEEE Aerospace and Electronic Systems Magazine*, 15(1):5–13, 2000. 51, 87

Sumit CHOPRA, Raia HADSELL et Yann LECUN : Learning a similarity metric discriminatively, with application to face verification. *In IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 539–546, 2005. 101

Gianluigi CIOCCA, Claudio CUSANO et Raimondo SCHETTINI : Image orientation detection using lbp-based features and logistic regression. *Multimedia Tools and Applications*, 74(9):3013–3034, 2015. 25

Djork-Arné CLEVERT, Thomas UNTERTHINER et Sepp HOCHREITER : Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv :1511.07289*, 2015. 39

Filipe COELHO et Cristina RIBEIRO : Evaluation of global descriptors for multimedia retrieval in medical applications. *In Workshop on Database and Expert Systems Applications (DEXA)*, pages 127–131, 2010. 14

Marius CORDTS, Mohamed OMRAN, Sebastian RAMOS, Timo REHFELD, Markus ENZWEILER, Rodrigo BENENSON, Uwe FRANKE, Stefan ROTH et Bernt SCHIELE : The cityscapes dataset for semantic urban scene understanding. *In IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3213–3223, Las Vegas, USA, juin 2016. vii, 50

Corinna CORTES et Vladimir VAPNIK : Support-vector networks. *Machine learning*, 20 (3):273–297, 1995. 7, 24, 27

Michael CRAMER : The dgpf-test on digital airborne camera evaluation - overview and test design. *Photogrammetrie-Fernerkundung-Geoinformation*, 2010(2):73–82, 2010. 7

David Richard CUTLER, Thomas C. EDWARDS, Karen H. BEARD, Adele CUTLER, Kyle T. HESS, Jacob GIBSON et Joshua J. LAWLER : Random forests for classification in ecology. *Ecology*, 88(11):2783–2792, 2007. 29

Navneet DALAL et Bill TRIGGS : Histograms of oriented gradients for human detection. *In IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893, San Diego, USA, June 2005. 21, 22

Michele DALPONTE, Lorenzo BRUZZONE et Damiano GIANELLE : Tree species classification in the southern alps based on the fusion of very high geometrical resolution multispectral/hyperspectral images and lidar data. *Remote Sensing of Environment*, 123:258–270, 2012. 51

Sukhendu DAS, T.T. MIRNALINEE et Koshy VARGHESE : Use of salient features for the design of a multistage framework to extract roads from high-resolution multispectral satellite images. *IEEE Geoscience and Remote Sensing Letters*, 49(10):3906–3931, 2011. 60

Jeffrey DEAN, Greg CORRADO, Rajat MONGA, Kai CHEN, Matthieu DEVIN, Mark MAO, Andrew SENIOR, Paul TUCKER, Ke YANG, Quoc V LE *et al.* : Large scale distributed deep networks. *In Advances in Neural Information Processing Systems (NIPS)*, pages 1223–1231, 2012. 37

Wei DI, Lei ZHANG, David ZHANG et Quan PAN : Studies on hyperspectral face recognition in visible spectrum with feature band selection. *IEEE Transactions on Systems, Man, and Cybernetics-Part A : Systems and Humans*, 40(6):1354–1361, 2010. 52

Jeff DONAHUE, Yangqing JIA, Oriol VINYALS, Judy HOFFMAN, Ning ZHANG, Eric TZENG et Trevor DARRELL : Decaf : A deep convolutional activation feature for generic visual recognition. *In International Conference on Machine Learning (ICML)*, pages 647–655, Beijing, China, juin 2014. vii, 41

John DUCHI, Elad HAZAN et Yoram SINGER : Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul): 2121–2159, 2011. 37

Aysegul DUNDAR et Ignacio GARCIA-DORADO : Context augmentation for convolutional neural networks. *arXiv preprint arXiv :1712.01653*, 2017. 83

Jan ELSEBERG, Dorit BORRMANN et Andreas NÜCHTER : Full wave analysis in 3D laser scans for vegetation detection in urban environments. *In Information, Communication*

and Automation Technologies (ICAT), 2011 XXIII International Symposium on, pages 1–7, 2011. 87

Hilal ERGUN, Yusuf Caglar AKYUZ, Mustafa SERT et Jianquan LIU : Early and late level fusion of deep convolutional neural networks for visual concept recognition. *International Journal of Semantic Computing*, 10(03):379–397, 2016. 77

Rong-En FAN, Kai-Wei CHANG, Cho-Jui HSIEH, Xiang-Rui WANG et Chih-Jen LIN : Liblinear : A library for large linear classification. *Journal of Machine Learning Research*, 9 (Aug):1871–1874, 2008. 25, 27

Zhengzheng FANG, Wei LI, Jinyi ZOU et Qian DU : Using CNN-based high-level features for remote sensing scene classification. *In IEEE International Geoscience and Remote Sensing Symposium*, pages 2610–2613, 2016. 72

Clement FARABET, Camille COUPRIE, Laurent NAJMAN et Yann LECUN : Learning hierarchical features for scene labeling. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 35(8):1915–1929, 2013. 49

Jason FARQUHAR, David HARDOON, Hongying MENG, John S. SHAWE-TAYLOR et Sandor SZEDMAK : Two view learning : Svm-2k, theory and practice. *In Advances in Neural Information Processing Systems (NIPS)*, pages 355–362, 2006. 29

Pedro F. FELZENSZWALB et Daniel P. HUTTENLOCHER : Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004. 49

Kunihiko FUKUSHIMA : Cognitron : A self-organizing multilayered neural network. *Biological Cybernetics*, 20(3-4):121–136, 1975. 39

Kunihiko FUKUSHIMA et Sei MIYAKE : Neocognitron : A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern Recognition*, 15(6):455 – 469, 1982. 39

Jing GAO et Pang-Ning TAN : Converting output scores from outlier detection algorithms into probability estimates. *In International Conference on Data Mining (ICDM)*, pages 212–221, Hong Kong, China, décembre 2006. 25

Junbin GAO et Christopher J. HARRIS : Some remarks on kalman filters for the multisensor fusion. *Information Fusion*, 3(3):191–201, 2002. 51

Christophe GARCIA et Manolis DELAKIS : Convolutional face finder : A neural architecture for fast and robust face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1408–1423, 2004. 57

Muriel GASTAUD : *Modèles de contours actifs pour la segmentation d'images et de vidéos*. Thèse de doctorat, Université Nice Sophia Antipolis, 2005. vii, 48

Jon GAUTHIER : Conditional generative adversarial nets for convolutional face generation. *Class Project for Stanford CS231N : Convolutional Neural Networks for Visual Recognition, Winter semester*, 2014(5):2, 2014. 53

Theo GEVERS, Joost VAN DE WEIJER et Harro STOKMAN : Color feature detection. *In* Rastislav LUKAC et Konstantinos N. PLATANIOTIS, éditeurs : *Color Image Processing : Methods and Applications*, volume 9, pages 203–226. CRC press, octobre 2006. 15

Ross GIRSHICK : Fast R-CNN. *In IEEE International Conference on Computer Vision (ICCV)*, Las Condes, Chile, décembre 2015. 46

Ross GIRSHICK, Jeff DONAHUE, Trevor DARRELL et Jitendra MALIK : Rich feature hierarchies for accurate object detection and semantic segmentation. *In IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, Columbus, USA, juin 2014. 46

Xavier GLOROT et Yoshua BENGIO : Understanding the difficulty of training deep feedforward neural networks. *In International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 9, pages 249–256, Sardinia, Italy, mai 2010. 33, 38

Grigorijs GOLDBERGS, Stefan W. MAIER, Shaun R. LEVICK et Andrew EDWARDS : Efficiency of individual tree detection approaches based on light-weight and low-cost uas imagery in australian savannas. *Remote Sensing*, 10(2), 2018. 49

Ian GOODFELLOW, Jean POUGET-ABADIE, Mehdi MIRZA, Bing XU, David WARDE-FARLEY, Sherjil OZAIR, Aaron COURVILLE et Yoshua BENGIO : Generative adversarial nets. *In Advances in Neural Information Processing Systems (NIPS)*, pages 2672–2680, 2014. 53

Hans P. GRAF, Eric COSATTO, Leon BOTTOU, Igor DOURDANOVIC et Vladimir VAPNIK : Parallel support vector machines : The cascade svm. *In Advances in Neural Information Processing Systems (NIPS)*, pages 521–528, 2005. 28

David GRANGIER, Léon BOTTOU et Ronan COLLOBERT : Deep convolutional networks for scene parsing. *In International Conference on Machine Learning (ICML) Deep Learning Workshop*, volume 3, Montreal, Canada, juin 2009. 49

Kristen GRAUMAN et Trevor DARRELL : The pyramid match kernel : discriminative classification with sets of image features. *In IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1458–1465 Vol. 2, Beijing, China, octobre 2005. 42

Jiuxiang GU, Zhenhua WANG, Jason KUEN, Lianyang MA, Amir SHAHROUDY, Bing SHUAI, Ting LIU, Xingxing WANG, Gang WANG, Jianfei CAI *et al.* : Recent advances in convolutional neural networks. *Pattern Recognition*, 2017. 45

Jingbo GUO, Jie XU, Songtao LIU, Di HUANG et Yunhong WANG : Occlusion-robust face detection using shallow and deep proposal based faster R-CNN. *In Chinese Conference on Biometric Recognition (CCBR)*, pages 3–12, Chengdu, China, octobre 2016. 46

David Lee HALL : *Mathematical techniques in multisensor data fusion.* Artech House, 1992. 51

David Lee HALL, Robert LINN et James LLINAS : A survey of data fusion systems. *In Data Structures and Target Classification*, volume 1470, pages 13–36, 1991. 51

David Lee HALL et James LLINAS : A challenge for the data fusion community i : research imperatives for improved processing. *In Symposium on Sensor Fusion*, 1994. 51

Caner HAZIRBAS, Lingni MA, Csaba DOMOKOS et Daniel CREMERS : Fusenet : Incorporating depth into semantic segmentation via fusion-based CNN architecture. *In Asian Conference on Computer Vision (ACCV)*, pages 213–228, 2016.

Haiqing HE, Min CHEN, Ting CHEN et Dajun LI : Matching of remote sensing images with complex background variations via siamese convolutional neural network. *Remote Sensing*, 10(2):355, 2018. 101

Kaiming HE, Georgia GKIOXARI, Piotr DOLLÁR et Ross GIRSHICK : Mask r-cnn. *In IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, Venice, Italy, octobre 2017a. xi, 102, 103

Kaiming HE, Xiangyu ZHANG, Shaoqing REN et Jian SUN : Spatial pyramid pooling in deep convolutional networks for visual recognition. *In* David FLEET, Tomas PAJDLA, Bernt SCHIELE et Tinne TUYTELAARS, éditeurs : *European Conference on Computer Vision (ECCV)*, pages 346–361, Zurich, Switzerland, août 2014. 42

Kaiming HE, Xiangyu ZHANG, Shaoqing REN et Jian SUN : Delving deep into rectifiers : Surpassing human-level performance on ImageNet classification. *In IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, Las Condes, Chile, décembre 2015. 39

Yaqian HE, Eungul LEE et Timothy A WARNER : A time series of annual land use and land cover maps of china from 1982 to 2013 generated using avhrr gimms ndvi3g data. *Remote Sensing of Environment*, 199:201–217, 2017b. 29

Ihsen HEDHLI, Gabriele MOSER, Sebastiano B. SERPICO et Josiane ZERUBIA : Classification of multisensor and multiresolution remote sensing images through hierarchical markov random fields. *IEEE Geoscience Remote Sensing Letters*, 14(12):2448–2452, 2017. 51

Geoffrey E. HINTON, Simon OSINDERO et Yee-Whye TEH : A fast learning algorithm for deep belief nets. *Neural Compututation*, 18(7):1527–1554, July 2006. 6

Tin Kam HO : Random decision forests. *In Proceedings of the International Conference on Document Analysis and Recognition*, volume 1, pages 278–282, Montreal, Canada, août 1995. 29

Tin Kam HO : The random subspace method for constructing decision forests. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998. 29, 30

Saihui HOU, Xu LIU et Zilei WANG : Dualnet : Learn complementary features for image recognition. *In IEEE International Conference on Computer Vision (ICCV)*, pages 502–510, Venice, Italy, octobre 2017. ix, 88, 89

Cho-Jui HSIEH, Si SI et Inderjit DHILLON : A divide-and-conquer solver for kernel support vector machines. *In International Conference on Machine Learning (ICML)*, pages 566–574, Beijing, China, juin 2014. 29

Jie HU, Li SHEN et Gang SUN : Squeeze-and-excitation networks. *arXiv preprint arXiv :1709.01507*, 2017. 32

Chengquan HUANG, Larry DAVIS et John TOWNSHEND : An assessment of support vector machines for land cover classification. *International Journal of Remote Sensing*, 23(4): 725–749, 2002. 25

Lin-Lin HUANG, Akinobu SHIMIZU, Yoshihoro HAGIHARA et Hidefumi KOBATAKE : Gradient feature extraction for classification-based face detection. *Pattern Recognition*, 36 (11):2501 – 2511, 2003. 15, 16

Yi HUANG et Adams Wai Kin KONG : Using a CNN ensemble for detecting pornographic and upskirt images. *In IEEE International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pages 1–7, New York, USA, September 2016. 78

Sergey IOFFE et Christian SZEGEDY : Batch normalization : Accelerating deep network training by reducing internal covariate shift. *In International Conference on Machine Learning (ICML)*, pages 448–456, Lille, France, juillet 2015. 44

Gareth JAMES, Daniela WITTEN, Trevor HASTIE et Robert TIBSHIRANI : *An introduction to statistical learning*, volume 112. Springer, 2013. 30

Hyeok JANG, Hun-Jun YANG, Dong-Seok JEONG et Hun LEE : Object classification using CNN for video traffic detection system. *In Workshop on Frontiers of Computer Vision (FCV)*, pages 1–4, Mokpo, South Korea, janvier 2015. 46

Kevin JARRETT, Korray KAVUKCUOGLU, Marc'Aurelio RANZATO et Yann LECUN : What is the best multi-stage architecture for object recognition ? *In IEEE International Conference on Computer Vision (ICCV)*, pages 2146–2153, Kyoto, Japan, September 2009. 40, 43

Carl F. JORDAN : Derivation of leaf-area index from quality of light on the forest floor. *Ecology*, 50(4):663–666, 1969. 75

He KAIMING, Zhang XIANGYU, Ren SHAOQING et Sun JIAN : Deep residual learning for image recognition. *IEEE conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, USA*, pages 770–778, juin 2016. 32, 45, 50

Rudolph Emil KALMAN : A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960. 51

Andrei V. KANAEV, Bobby J. DANIEL, József Gáti John von NEUMANN, Angela M. KIM et Kenneth R. LEE : Object level hsi-lidar data fusion for automated detection of difficult targets. *Optics express*, 19(21):20916–20929, 2011. 51, 87

Shahid KARIM, Ye ZHANG, Saad ALI et Muhammad Rizwan ASIF : An improvement of vehicle detection under shadow regions in satellite imagery. *In International Conference on Graphic and Image Processing (ICGIP)*, volume 10615, page 106154D, Chengdu, China, décembre 2018. 23

Michael KASS, Andrew WITKIN et Demetri TERZOPOULOS : Snakes : Active contour models. *International Journal of Computer Vision*, 1988. 48

Csaba KERTÉSZ et Vincit OY : Texture-based foreground detection. 2011. 14

Mahdi KHODADADZADEH, Jun LI, Saurabh PRASAD et Antonio PLAZA : Fusion of hyperspectral and lidar remote sensing data using multiple feature learning. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 8(6):2971–2983, 2015. 87

Diederik P. KINGMA et Jimmy BA : Adam : A method for stochastic optimization. *arXiv preprint arXiv :1412.6980*, 2014. 37

Diederik P. KINGMA et Max WELLING : Auto-encoding variational bayes. *arXiv preprint arXiv :1312.6114*, 2013. 53

Les KITCHEN et Azriel ROSENFELD : Gray-level corner detection. *Pattern Recognition Letters*, 1(2):95–102, 1982. 45

Iasonas KOKKINOS : Ubernet : Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. *In IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, page 8, Honolulu, Hawaii, juillet 2017. 88, 90

Ulrich KRESSEL : Advances in kernel methods-support vector learning, chapter 15-pairwise classification and support vector machines, 1999. 28

FJ KRIEGLER, WA MALILA, RF NALEPKA et W RICHARDSON : Preprocessing transformations and their effects on multispectral recognition. *In Remote Sensing of Environment*, page 97, 1969. 75

Aalex KRIZHEVSKÝ, Ilya SUTSKEVER et Geoffrey E. HINTON : ImageNet classification with deep convolutional neural networks. *In Conference on Neural Information Processing Systems (NIPS)*, pages 1097–1105, Lake Tahoe, USA, décembre 2012. viii, 6, 32, 38, 43, 46, 56, 60, 61, 72, 78

John LAFFERTY, Andrew MCCALLUM et Fernando CN PEREIRA : Conditional random fields : Probabilistic models for segmenting and labeling sequence data. 2001. 49

Adrien LAGRANGE, Bertrand LE SAUX, Anne BEAUPERE, Alexandre BOULCH, Adrien CHAN-HON-TONG, Stéphane HERBIN, Hicham RANDRIANARIVO et Marin FERECATU : Benchmarking classification of earth-observation data : From learning explicit features to convolutional networks. *In International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 4173–4176, 2015. 51

Kevin LAI, Liefeng BO, Xiaofeng REN et Dieter FOX : A large-scale hierarchical multi-view rgb-d object dataset. *In IEEE International Conference on Robotics and Automation (ICRA)*, pages 1817–1824, Shanghai, China, mai 2011. 52

Svetlana LAZEBNIK, Cordelia SCHMID et Jean PONCE : Beyond bags of features : Spatial pyramid matching for recognizing natural scene categories. *In IEEE conference on Computer Vision and Pattern Recognition (CVPR), New York, USA*, volume 2, pages 2169–2178, juin 2006. 42

Yann LECUN, Bernhard E. BOSER, John S. DENKER, Donnie HENDERSON, Richard E. HOWARD, Wayne E. HUBBARD et Lawrence D. JACKEL : Handwritten digit recognition with a back-propagation network. *In Advances in Neural Information Processing Systems (NIPS)*, pages 396–404, 1990. 6

Yann LECUN, Léon BOTTOU, Yoshua BENGIO et Patrick HAFFNER : Gradient-based learning applied to document recognition. November 1998. 6

Yann LECUN, Larry D. JACKEL, Léon BOTTOU, Corinna CORTES, John S. DENKER, Harris DRUCKER, Isabelle GUYON, Urs A. MULLER, Eduard SACKINGER, Patrice SIMARD *et al.* : Learning algorithms for classification : A comparison on handwritten digit recognition. *Neural networks : the statistical mechanics perspective*, 261:276, 1995. 39, 72

Jerome LETTVIN, Humberto MATURANA, Warren S. MCCULLOCH et Walter PITTS : What the frog's eye tells the frog's brain. *In Institute of Radio Engineers*, volume 47, pages 1940–1951, Nov 1959. 32

Wenzhi LIAO, Rik BELLENS, Aleksandra PIZURICA, Sidharta GAUTAMA et Wilfried PHILIPS : Graph-based feature fusion of hyperspectral and lidar remote sensing data using morphological features. *In Proceedings of the IEEE International Geoscience and Remote Sensing Symposium*, volume 7, pages 4942–4945, 2013. 87

Ee Hui LIM et David SUTER : 3D terrestrial LIDAR classifications with super-voxels and multi-scale conditional random fields. *Computer-Aided Design*, 41(10):701–710, 2009. 87

Qingshan LIU, Renlong HANG, Huihui SONG et Zhi LI : Learning multiscale deep features for high-resolution satellite image scene classification. *IEEE Geoscience and Remote Sensing*, 56(1):117–126, 2018. 5

Xiaolong LIU et Yanchen BO : Object-based crop species classification based on the combination of airborne hyperspectral images and lidar data. *Remote Sensing*, 7(1):922–950, 2015. 51

Yanfei LIU, Yanfei ZHONG, Feng FEI et Liangpei ZHANG : Scene semantic classification based on random-scale stretched convolutional neural network for high-spatial resolution remote sensing imagery. *In IEEE International Geoscience and Remote Sensing Symposium*, pages 763–766, 2016. 72

Yansong LIU, Sankaranarayanan PIRAMANAYAGAM, Sildomar T MONTEIRO et Eli SABER : Dense semantic labeling of very-high-resolution aerial imagery and LiDAR with fully-convolutional neural networks and higher-order CRFs. *In IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, Hawaii, juillet 2017. 51

James LLINAS et David Lee HALL : A challenge for the data fusion community ii : Infrastructure imperatives. *In Symposium on Sensor Fusion*, volume 16, 1994. 51

David G. LOWE : Object recognition from local scale-invariant features. *In IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1150–1157, Osaka, Japan, décembre 1999. 17

David G. LOWE : Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. vi, 17, 19, 20, 22

Siwei LYU et Eero P SIMONCELLI : Nonlinear image representation using divisive normalization. *In IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, Anchorage, USA, June 2008. 43

Laurens van der MAATEN et Geoffrey HINTON : Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008. vii, 41

Sabelo MADONSELA, Moses Azong CHO, Abel RAMOELO, Onisimo MUTANGA et Laven NAIDOO : Estimating tree species diversity in the savannah using NDVI and woody canopy cover. *International Journal of Applied Earth Observation and Geoinformation*, 66:106–115, 2018. 76

Emmanuel MAGGIORI, Yuliya TARABALKA, Guillaume CHARPIAT et Pierre ALLIEZ : High-resolution aerial image labeling with convolutional neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 55(12):7092–7103, 2017. 87

Adhiyaman MANICKAM, Ezhilmaran DEVARASAN, Gunasekaran MANOGARAN, Malarvizhi Kumar PRIYAN, R VARATHARAJAN, Ching-Hsien HSU et Raja KRISHNAMOORTHI : Score level based latent fingerprint enhancement and matching using sift feature. *Multimedia Tools and Applications*, pages 1–21, 2018. 21

Kevis-Kokitsi MANINIS, Jordi PONT-TUSET, Pablo ARBELÁEZ et Luc VAN GOOL : Convolutional oriented boundaries : From image segmentation to high-level tasks. *IEEE Transactions on Tattern Analysis and Machine Intelligence (TPAMI)*, 40(4):819–833, 2018. 72

David R. MARTIN, Charless C. FOWLKES et Jitendra MALIK : Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 26(5):530–549, 2004. 47

Warren S. MCCULLOCH et Walter PITTS : A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943. 38

Adele MEHRANFAR, Nasser GHADIRI, Morteza KOUHSAR et Ashkan GOLSHANI : A type-2 fuzzy data fusion approach for building reliable weighted protein interaction networks with application in protein complex detection. *Computers in Biology and Medicine*, 88:18–31, 2017.

George E. MEYER : Machine vision identification of plants. *Recent Trends for Enhancing the Diversity and Quality of Soybean Products. Krezhova D (ed.) Croatia : InTech*, 2011. 76

Cyrille MIGNIOT : *Segmentation de personnes dans les images et les videos.* Thèse de doctorat, Université Grenoble Alpes, 2012. 23

Emilie MORVANT, Amaury HABRARD et Stéphane AYACHE : Majority vote of diverse classifiers for late fusion. *In Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 153–162, 2014. 76

Giorgos MOUNTRAKIS, Jungho IM et Caesar OGOLE : Support vector machines in remote sensing : A review. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(3):247–259, 2011. 25

Robin R. MURPHY : Sensor fusion. *The Handbook of Brain Theory and Neural Networks*, 1994.

Vinod NAIR et Geoffrey E. HINTON : Rectified linear units improve restricted boltzmann machines. *In International Conference on Machine Learning (ICML)*, pages 807–814, Haifa, Israel, juin 2010. 38, 60

Nasser M. NASRABADI : Pattern recognition and machine learning. *Journal of Electronic Imaging*, 16(4):049901, 2007. 24

Jiquan NGIAM, Aditya KHOSLA, Mingyu KIM, Juhan NAM, Honglak LEE et Andrew Y NG : Multimodal deep learning. *In International Conference on Machine Learning (ICML)*, pages 689–696, Bellevue, USA, juin 2011. 51

Feiping NIE, Xiaoqian WANG et Heng HUANG : Multiclass capped lp-norm svm for robust classifications. *In Conference on Artificial Intelligence (AAAI)*, San Francisco, USA, février 2017. 29

Hitoshi NIIGAKI, Jun SHIMAMURA et Masashi MORIMOTO : Circular object detection based on separability and uniformity of feature distributions using bhattacharyya coefficient. *In International Conference on Pattern Recognition (ICPR)*, pages 2009–2012, Tsukuba, Japan, novembre 2012. 13

Hyeonwoo NOH, Seunghoon HONG et Bohyung HAN : Learning deconvolution network for semantic segmentation. *In IEEE International Conference on Computer Vision (ICCV)*, pages 1520–1528, Las Condes, Chile, décembre 2015. 89, 92, 97

Aude OLIVA et Antonio TORRALBA : Modeling the shape of the scene : A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, 2001. vii, 41

Stanley OSHER et James A. SETHIAN : Fronts propagating with curvature-dependent speed : algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, 79(1):12–49, 1988. 49

Sakrapee PAISITKRIANGKRAI, Jamie SHERRAH, Pranam JANNEY, Van-Den HENGEL *et al.* : Effective semantic pixel labelling with convolutional networks and conditional random fields. *In IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 36–43, Boston, USA, juin 2015. 51

Mahesh PAL : Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, 26(1):217–222, 2005. 25, 29

Mahesh PAL et Giles M. FOODY : Evaluation of svm, rvm and smlr for accurate image classification with limited ground data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 5(5):1344–1355, 2012. 25

George PAPANDREOU, Iasonas KOKKINOS et Pierre-André SAVALLE : Modeling local and global deformations in deep learning : Epitomic convolution, multiple instance learning, and sliding window detection. *In IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 390–399, Boston, USA, juin 2015. 40

Jérôme PASQUET, Marc CHAUMONT, Gérard SUBSOL et Mustapha DERRAS : An efficient multi-resolution SVM network approach for object detection in aerial images. *In International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, Boston, USA, septembre 2015. 4

Jérôme PASQUET, Thibault DESERT, Olivier BARTOLI, Marc CHAUMONT, Carole DELENNE, Gérard SUBSOL, Mustapha DERRAS et Nanée CHAHINIAN : Detection of manhole covers in high-resolution aerial images of urban areas by combining two methods. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 9(5):1802–1807, 2016. 4, 13, 23

Deepak PATHAK, Philipp KRAHENBUHL, Jeff DONAHUE, Trevor DARRELL et Alexei A. EFROS : Context encoders : Feature learning by inpainting. *In IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2536–2544, Las Vegas, USA, juin 2016. 53

Florin Alexandru PAVEL, Zhiyong WANG et David Dagan FENG : Reliable object recognition using sift features. *In IEEE International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6, 2009. 21

Fabian PEDREGOSA, Gaël VAROQUAUX, Alexandre GRAMFORT, Vincent MICHEL, Bertrand THIRION, Olivier GRISEL, Mathieu BLONDEL, Peter PRETTENHOFER, Ron WEISS, Vincent DUBOURG *et al.* : Scikit-learn : Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011. 67

Stefan PETSCHARNIG, Klaus SCHÖFFMANN, Jenny BENOIS-PINEAU, Souad CHAABOUNI et Jörg KECKSTEIN : Early and late fusion of temporal information for classification of surgical actions in laparoscopic gynecology. *In IEEE International Symposium on Computer-Based Medical Systems (CBMS)*, pages 369–374, Karlstad, Sweden, juin 2018. 51

Maxime PORTAZ, Matthias KOHL, Jean-Pierre CHEVALLET, Georges QUÉNOT et Philippe MULHEM : Object instance identification with fully convolutional networks. *Multimedia Tools and Applications*, pages 1–18, 2018. 46

Judith M. S. PREWITT : Object enhancement and extraction. *Picture processing and Psychopictorics*, 10(1):15–19, 1970. 47

Sheng QIAN, Hua LIU, Cheng LIU, Si WU et Hau San WONG : Adaptive activation functions in convolutional neural networks. *Neurocomputing*, 272:204 – 212, 2018. ISSN 0925-2312. 39

Zhu QIANG, Yeh MEI-CHEN, Cheng KWANG-TING et Shai AVIDAN : Fast human detection using a cascade of histograms of oriented gradients. *In IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 1491–1498, New York, USA, juin 2006. 22, 56, 66, 71

John Ross QUINLAN : Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986. 29

Suzaimah RAMLI, Mohd Marzuki MUSTAFA, Aini HUSSAIN et Dzuraidah Abdul WAHAB : Histogram of intensity feature extraction for automatic plastic bottle recycling system using machine vision. *American Journal of Environmental Sciences*, 4(6):583–588, 2008. 15

Shaoqing REN, Kaiming HE, Ross GIRSHICK et Jian SUN : Faster R-CNN : Towards real-time object detection with region proposal networks. *In Neural Information Processing Systems (NIPS)*, 2015. 47

Xiaofeng REN : Multi-scale improves boundary detection in natural images. *In European Conference on Computer Vision (ECCV)*, pages 533–545, Munich, Germany, septembre 2008. 48

Danilo Jimenez REZENDE, Shakir MOHAMED et Daan WIERSTRA : Stochastic back-propagation and approximate inference in deep generative models. *arXiv preprint arXiv :1401.4082*, 2014. 53

Lawrence G. ROBERTS : *Machine perception of three-dimensional solids*. Thèse de doctorat, Massachusetts Institute of Technology, 1963. 47

John W. ROUSE JR, Richard H. HAAS, John A. SCHELL et Donald W. DEERING : Monitoring vegetation systems in the great plains with erts. 1974. 75

Ethan RUBLEE, Vincent RABAUD, Kurt KONOLIGE et Gary BRADSKI : Orb : An efficient alternative to sift or surf. *In IEEE International Conference on Computer Vision (ICCV)*, pages 2564–2571, Barcelona, Spain, novembre 2011. 21

David E. RUMELHART, Geoffrey E. HINTON et Ronald J. WILLIAMS : Learning representations by back-propagating errors. *nature*, 323(6088):533, 1986. 33, 34

David SAAD : Online algorithms and stochastic approximations. *Online Learning*, 5, 1998. 37

Dominik SCHERER, Andreas MÜLLER et Sven BEHNKE : Evaluation of pooling operations in convolutional architectures for object recognition. *In International Conference on Artificial Neural Networks (ICANN)*, pages 92–101, Thessaloniki, Greece, septembre 2010. 42

Michael SCHMITT et Xiao Xiang ZHU : Data fusion and remote sensing : An ever-growing relationship. *IEEE Geoscience and Remote Sensing Magazine*, 4(4):6–23, 2016. 51, 74

Bernhard SCHÖLKOPF, Alex J. SMOLA, Robert C. WILLIAMSON et Peter L. BARTLETT : New support vector algorithms. *Neural Computation*, 12(5):1207–1245, 2000. 28

Szabolcs SERGYAN : Color histogram features based image classification in content-based image retrieval systems. *In International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pages 221–224, 2008. 15

Pierre SERMANET, David EIGEN, Xiang ZHANG, Michael MATHIEU, Rob FERGUS et Yann LECUN : Overfeat : Integrated recognition, localization and detection using convolutional networks. *In International Conference on Learning Representations (ICLR)*, Banff, Canada, April 2014. 56, 59

Huanfeng SHEN, Xinghua LI, Qing CHENG, Chao ZENG, Gang YANG, Huifang LI et Liangpei ZHANG : Missing information reconstruction of remote sensing data : A technical review. *IEEE Geoscience and Remote Sensing Magazine*, 3(3):61–85, 2015. 87

Karen SIMONYAN et Andrew ZISSERMAN : Very deep convolutional networks for large-scale image recognition. *In Proceedings of the International Conference on Learning Representation (ICLR)*, Toulon, France, avril 2015. 32, 50

Cees G. M. SNOEK, Marcel WORRING, Jan-Mark GEUSEBROEK, Dennis C. KOELMA et Frank J. SEINSTRA : The mediamill trecvid 2004 semantic video search engine. *In TRECVID Workshop*, 2004. 76

Cees G. M. SNOEK, Marcel WORRING et Arnold W. M. SMEULDERS : Early versus late fusion in semantic video analysis. *In Proceedings of the 13th annual ACM international conference on Multimedia*, pages 399–402, Singapore, novembre 2005. ix, 51, 76, 77

Irvin SOBEL : An isotropic 3× 3 image gradient operator. *Machine Vision for Three-Dimensional Scenes*, pages 376–379, 1990. 16

Richard SOCHER, Brody HUVAL, Bharath BATH, Christopher D MANNING et Andrew Y NG : Convolutional-recursive deep learning for 3d object classification. *In Advances in Neural Information Processing Systems (NIPS)*, pages 656–664, 2012. 49

Mingjun SONG et Daniel CIVCO : Road extraction using svm and image segmentation. *Photogrammetric Engineering & Remote Sensing*, 70(12):1365–1371, 2004. 49

Jost Tobias SPRINGENBERG, Alexey DOSOVITSKIY, Thomas BROX et Martin RIEDMILLER : Striving for simplicity : The all convolutional net. *In arXiv :1412.6806, also appeared at ICLR 2015 Workshop Track*, San Diego, USA, mai 2015. 42

Nitish SRIVASTAVA, Geoffrey E. HINTON, Alex KRIZHEVSKY, Ilya SUTSKEVER et Ruslan SALAKHUTDINOV : Dropout : A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. 7

Xian SUN, Hongqi WANG et Kun FU : Automatic detection of geospatial objects using taxonomic semantics. *IEEE Geoscience and Remote Sensing Letters*, 7(1):23–27, 2010. 5, 60

Christian SZEGEDY, Wei LIU, Yangqing JIA, Pierre SERMANET *et al.* : Going deeper with convolutions. *In IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, Boston, USA, juin 2015. viii, 32, 39, 56, 60, 61, 62, 72

Demetri TERZOPOULOS, John PLATT, Alan BARR et Kurt FLEISCHER : Elastically deformable models. *ACM SIGGRAPH Computer Graphics*, 1987. 48

Luan TRAN, Xiaoming LIU, Jiayu ZHOU et Rong JIN : Missing modalities imputation via cascaded residual autoencoder. *In IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1405–1414, Honolulu, Hawaii, juillet 2017. 52, 87

Compton J. TUCKER : Red and photographic infrared linear combinations for monitoring vegetation. *Remote Sensing of Environment*, 8(2):127–150, 1979. 75

Paul E. UTGOFF : Incremental induction of decision trees. *Machine Learning*, 4(2):161–186, 1989. 29

Paul E. UTGOFF : An improved algorithm for incremental induction of decision trees. *In Machine Learning Proceedings*, pages 318–325. Elsevier, 1994. 29

Vladimir VAPNIK : *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995. 7, 24

Vladimir VAPNIK : *Statistical learning theory*. Wiley, New York, 1998. 27

Paul VIOLA et Michael JONES : Rapid object detection using a boosted cascade of simple features. *In IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages I–511–I–518 vol.1, Kauai, USA, décembre 2001. 4

Michele VOLPI et Devis TUIA : Dense semantic labeling of subdecimeter resolution images with convolutional neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 55(2):881–893, 2017. 87

Jörg WAGNER, Volker FISCHER, Michael HERMAN et Sven BEHNKE : Multispectral pedestrian detection using deep fusion convolutional neural networks. *In European Symposium on Artificial Neural Networks (ESANN)*, Bruges, Belgium, April 2016. vii, 51, 52, 78

Kiri L. WAGSTAFF, You LU, Alice STANBOLI, Kevin GRIMES, Thamme GOWDA et Jordan PADAMS : Deep mars : Cnn classification of mars imagery for the pds imaging atlas. *In Conference on Innovative Applications of Artificial Intelligence (IAAI)*, New Orleans, USA, février 2018. 72

Edward WALTZ : Data fusion for c3i : A tutorial. *Command, Control, Communications Intelligence (C3I) Handbook*, pages 217–226, 1986. 51

Edward WALTZ et James LLINAS : *Multisensor data fusion*, volume 685. Artech house Boston, 1990. 51, 87

Hongzhou WANG, Craig GLENNIE et Saurabh PRASAD : Voxelization of full waveform lidar data for fusion with hyperspectral imagery. *In International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 3407–3410, 2013. 87

Jinjun WANG, Jianchao YANG, Kai YU, Fengjun LV, Thomas HUANG et Yihong GONG : Locality-constrained linear coding for image classification. *In IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3360–3367, San Francisco, USA, juin 2010. vii, 41

Thijs WESTERVELD, Arjen P. DE VRIES, Alex van BALLEGOOIJ, Franciska de JONG et Djoerd HIEMSTRA : A probabilistic multimedia retrieval model and its evaluation. *EURASIP Journal on Advances in Signal Processing*, 2003(2):985676, 2003. 76

Jason WESTON, Chris WATKINS *et al.* : Support vector machines for multi-class pattern recognition. *In In European Symposium on Artificial Neural Networks Computational Intelligence and Machine Learning (ESANN)*, volume 99, pages 219–224, 1999. 28

Ren WU, Shengen YAN, Yi SHAN, Qingqing DANG et Gang SUN : Deep image : Scaling up image recognition. *arXiv preprint arXiv :1501.02876*, 7(8), 2015. 45

Ting-Fan WU, Chih-Jen LIN et Ruby C WENG : Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5(Aug):975–1005, 2004. 76

Junyuan XIE, Linli XU et Enhong CHEN : Image denoising and inpainting with deep neural networks. *In Advances in Neural Information Processing Systems (NIPS)*, pages 341–349, 2012. 53

Guanshuo XU, Han-zhou WU et Yun Q. SHI : Ensemble of CNNs for steganalysis : An empirical study. *In Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*, IH&#38 ;MMSec '16, pages 103–107, New York, NY, USA, 2016. ACM. 78

Jian YANG, Yuhong HE et John CASPERSEN : Individual tree-based species classification for uneven-aged, mixed-deciduous forests using multi-seasonal worldview-3 images. *In International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 827–830, Fort Worth, USA, juillet 2017. 5

Fisher YU et Vladlen KOLTUN : Multi-scale context aggregation by dilated convolutions. *In Proceedings of the International Conference on Learning Representation (ICLR)*, San Diego, USA, mai 2015. 40

Jincheng YU, Kaiyuan GUO, Yiming HU, Xuefei NING, Jiantao QIU, Huizi MAO, Song YAO, Tianqi TANG, Boxun LI, Yu WANG *et al.* : Real-time object detection towards high power efficiency. *In Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 704–708, Dresden, Germany, mars 2018. IEEE. 72

Matthew ZEILER : Adadelta : an adaptive learning rate method. *arXiv preprint arXiv :1212.5701*, 2012. 37

Matthew ZEILER et Robert FERGUS : Stochastic pooling for regularization of deep convolutional neural networks. *In International Conference on Learning Representation (ICLR)*, Scottsdale, USA, mai 2013. 32

Qi-xing ZHANG, Gao-hua LIN, Yong-ming ZHANG, Gao XU et Jin-jun WANG : Wildland forest fire smoke detection based on faster r-cnn using synthetic smoke images. *Procedia engineering*, 211:441–446, 2018. 72

Wanceng ZHANG, Xian SUN et Kun FU : Object detection in high-resolution remote sensing images using rotation invariant parts based model. *IEEE Geoscience and Remote Sensing Letters*, 11(1):74–78, 2014. 60

Wanceng ZHANG, Xian SUN et Chenyuan WANG : A generic discriminative part-based model for geospatial object detection in optical remote sensing images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 99:30–44, 2015. 60

Zhong-Qiu ZHAO, Haiman BIAN, Donghui HU, Wenjuan CHENG et Hervé GLOTIN : Pedestrian detection based on fast R-CNN and batch normalization. *In International Conference on Intelligent Computing (ICIC)*, pages 735–746, Liverpool, UK, août 2017. 45

Chao ZHU et Xu-Cheng YIN : Effective human detection via multi-model classification and adaptive late fusion. *International Journal of Wavelets, Multiresolution and Information Processing (IJWMIP)*, 16(2):1–16, 2018. 76

Haigang ZHU, Xiaogang CHEN, Weiqun DAI, Kun FU, Qixiang YE et Jianbin JIAO : Orientation robust object detection in aerial images using deep convolutional neural network. *In IEEE International Conference on Image Processing (ICIP)*, pages 3735–3739, Québec, Canada, septembre 2015. 41