Apprentissage automatique pour la détection d'anomalies dans les données ouvertes : application à la cartographie

Rémi Delassus

▶ **To cite this version:**

Rémi Delassus. Apprentissage automatique pour la détection d'anomalies dans les données ouvertes : application à la cartographie. Traitement des images [eess.IV]. Université de Bordeaux, 2018. Français. NNT : 2018BORD0230 . tel-02100741

HAL Id: tel-02100741
https://theses.hal.science/tel-02100741

Submitted on 16 Apr 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE

PRÉSENTÉE À

L'UNIVERSITÉ DE BORDEAUX

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET D'INFORMATIQUE

par **Rémi Delassus**

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ : INFORMATIQUE

Apprentissage automatique pour la détection d'anomalies dans les données ouvertes : application à la cartographie

---

Date de soutenance : 23 Novembre 2018

Devant la commission d'examen composée de :

Guy Melançon ..	Professeur, Université de Bordeaux	Directeur
Romain Giot	Maître de conférences, Université de Bordeaux	Co-encadrant
Hubert Cardot ..	Professeur, Université de Tours	Rapporteur
Dino Ienco	Chargé de recherche (HDR), IRSTEA	Rapporteur
Thierry Urruty .	Maître de Conférences, Université de Poitiers .	Examinateur
Vincent Lepetit .	Professeur, Université de Bordeaux	Président du jury
Gabriele Barbieri	Directeur Scientifique, Qucit	Invité

– 2018 –

Titre Apprentissage automatique pour la détection d'anomalies dans les données ouvertes : application à la cartographie

Résumé Dans cette thèse nous étudions le problème de détection d'anomalies dans les données ouvertes utilisées par l'entreprise Qucit ; aussi bien les données métiers de ses clients, que celles permettant de les contextualiser. Dans un premier temps, nous nous sommes intéressés à la détection de vélos défectueux au sein des données de trajets du système de vélo en libre service de New York. Nous cherchons des données reflétant une anomalie dans la réalité. Des caractéristiques décrivant le comportement de chaque vélo observé sont partitionnés. Les comportements anormaux sont extraits depuis ce partitionnement et comparés aux rapports mensuels indiquant le nombre de vélos réparés ; c'est un problème d'apprentissage à sortie agrégée. Les résultats de ce premier travail se sont avérés insatisfaisant en raison de la pauvreté des données. Ce premier volet des travaux a ensuite laissé place à une problématique tournée vers la détection de bâtiments au sein d'images satellites. Nous cherchons des anomalies dans les données géographiques qui ne reflètent pas la réalité. Nous proposons une méthode de fusion de modèles de segmentation améliorant la métrique d'erreur jusqu'à +7% par rapport à la méthode standard. Nous évaluons la robustesse de notre modèle face à la suppression de bâtiments dans les étiquettes, afin de déterminer à quel point les omissions sont susceptibles d'en altérer les résultats. Ce type de bruit est communément rencontré au sein des données OpenStreetMap, régulièrement utilisées par Qucit, et la robustesse observée indique qu'il pourrait être corrigé.

Mots-clés Apprentissage Profond, Segmentation d'Images, Images Satellites

Title Satellite images analysis for anomaly detection in open geographical data

Abstract In this thesis we study the problem of anomaly detection in the open data used by the Qucit company, both the business data of its customers, as well as those allowing to contextualize them. We are looking for data that reflects an anomaly in reality. Initially, we were interested in detecting defective bicycles in the trip data of New York's bike share system. Characteristics describing the behaviour of each observed bicycle are clustered. Abnormal behaviors are extracted from this clustering and compared to monthly reports indicating the number of bikes repaired; this is an aggregate learning problem. The results of this first work were unsatisfactory due to the paucity of data. This first part of the work then gave way to a problem focused on the detection of buildings within satellite images. We are looking for anomalies in the geographical data that do not reflect reality. We propose a method of merging segmentation models that improves the error metric by up to +7% over the standard method. We assess the robustness of our model to the removal of buildings from labels to determine the extent to which omissions are likely to alter the results. This type of noise is commonly encountered within the OpenStreetMap data, regularly used by Qucit, and the robustness observed indicates that it could be corrected.

Keywords Deep Learning, Image Segmentation, Satellite Images

Laboratoire d'accueil Laboratoire Bordelais de Recherche en Informatique (LaBRI), Bâtiment A30, Domaine universitaire, 351, cours de la Libération, 33405 Talence

Remerciements

Il me sera difficile de remercier toutes les personnes qui ont su m'apporter leur soutien durant ces trois années. Je commencerai évidemment par remercier mon encadrant Romain Giot, tant pour sa disponibilité que pour ses précieux conseils, notamment lors de ses rigoureuses relectures. Je remercie également Guy Melançon d'avoir accepté de diriger ma thèse malgré des responsabilités administratives toujours plus nombreuses.

Je suis sincèrement reconnaissant envers Raphaël Cherrier, président et directeur de Qucit, d'avoir accepté de m'employer en tant que doctorant au sein de son entreprise qui n'était même pas encore créée lorsque nous sommes entrés en contact. Ses idées ont été motrices dans la réalisation de ces travaux. Cet emploi a aussi été en partie financé par la bourse CIFRE numéro 2014/1303 et je remercie pour cela l'Association Nationale de la Recherche et de la Technologie.

Au sein de Qucit, je tiens à remercier Gabriele Barbieri pour sa sincère préoccupation pour le bon déroulement de ma thèse. Je ne pense pas que j'aurais pu la mener à bien sans cela. Une pensée également pour Yassine Alouini qui m'a apporté son aide lorsqu'un article un peu trop mathématique me posait problème.

Je remercie mes collègues et amis du Laboratoire Bordelais de Recherche en Informatique, Christelle Al Hasrouty, Elyas Ben Hadj Yahia, Wei Chen, Hanyang Ciao, Mohamed Oumaziz, et Mohammed Senhaji, qui vivent ou ont vécu les mêmes épreuves, pour leur soutien mutuel.

Bien sûr, je ne serai pas sans citer ma famille, qui m'a toujours encouragé dans mes projets et offert les moyens de les réaliser. Je les remercie également de leur hospitalité lorsque j'ai eu besoin de m'isoler pour rédiger ce manuscrit.

Mes tendres remerciements enfin à Elody Adjei Owusu, qui m'a apporté son soutien et a su faire preuve d'une grande patience quand je ne savais plus trouver pour elle le temps qu'elle méritait.

À tous mes amis, à toute personne que je n'ai pas su citer ici et qui m'ont apporté de la joie, du support, de l'aide, quand j'en avais besoin, merci.

Table des matières

Table des matières **vii**

1 Introduction **3**
1.1 Contexte . . . 3
1.1.1 Détection de vélos défectueux . . . 4
1.1.2 Extraction de bâtiments au sein d'images satellites . . . 5
1.2 Contributions . . . 6
1.2.1 Détection de vélos défectueux . . . 6
1.2.2 Extraction de bâtiments au sein d'images satellites . . . 8
1.3 Organisation du manuscrit . . . 9

2 Apprentissage automatique **11**
2.1 Prérequis : mécanismes de l'apprentissage automatique . . . 11
2.1.1 Modélisation d'un problème d'apprentissage automatique . . . 12
2.1.2 Fonction de coût . . . 14
2.1.3 Optimisation des paramètres . . . 16
2.2 Apprentissage supervisé . . . 19
2.2.1 Sous-apprentissage et sur-apprentissage . . . 20
2.2.2 Régularisation . . . 22
2.2.3 Optimisation des hyper-paramètres . . . 23
2.3 Apprentissage profond . . . 26
2.3.1 Principes généraux . . . 26
2.3.2 Réseaux convolutifs . . . 35
2.4 Apprentissage non supervisé . . . 40
2.4.1 Partitionnement . . . 41
2.4.2 Détection d'anomalies . . . 42
2.4.3 Utilisation des Réseaux de neurones . . . 46
2.5 Application de l'apprentissage non supervisé au problème de détection de vélos défectueux au sein d'un système de vélo en libre service. . . . 48
2.5.1 État de l'art . . . 48
2.5.2 Méthode proposée . . . 49
2.5.3 Protocole Expérimental . . . 54
2.5.4 Résultats . . . 56
2.5.5 Conclusion . . . 59

3 Détection d'anomalies dans les systèmes de cartographie 61
3.1 Prérequis : apprentissage profond pour la segmentation d'image . . . 62
3.1.1 Analyse d'images pour la segmentation . . . 62
3.1.2 État de l'art des techniques d'apprentissage profond pour l'analyse d'image . . . 63
3.1.3 Segmentation d'instances . . . 74
3.1.4 État de l'art de la détection des contours de bâtiments dans les images satellites . . . 77
3.2 Challenges SpaceNet . . . 79
3.2.1 Données . . . 80
3.2.2 Évaluation . . . 82
3.3 Étude comparative des représentations de la vérité terrain . . . 83
3.3.1 Transformation des étiquettes . . . 83
3.3.2 Protocole expérimental . . . 84
3.3.3 Résultats . . . 87
3.3.4 Conclusion . . . 90
3.4 Fusion de modèles pour la segmentation de bâtiments . . . 91
3.4.1 État de l'art . . . 93
3.4.2 Combineurs proposés . . . 97
3.4.3 Protocole expérimental . . . 100
3.4.4 Résultats . . . 102
3.4.5 Conclusion . . . 109
3.5 Analyse de la résistance au bruit . . . 109
3.5.1 État de l'art . . . 110
3.5.2 Proposition de modèles renforcés . . . 112
3.5.3 Protocole expérimental . . . 115
3.5.4 Résultats . . . 117
3.5.5 Conclusion . . . 120

4 Conclusion 123
4.1 Récapitulatif des problèmes abordés . . . 123
4.1.1 Les problématiques de Qucit . . . 123
4.1.2 Contributions . . . 124
4.2 Perspectives . . . 125
4.2.1 Approfondissement des études . . . 125
4.2.2 Modèle plus performant . . . 126
4.2.3 Outil de détection d'erreurs OSM . . . 127
4.2.4 Application à d'autres problèmes . . . 127

Bibliographie 131

Notations

Nous définissons ici les différentes notations rencontrées dans le manuscrit.

Dimensions

a	Un scalaire
$\boldsymbol{a}$	Un vecteur
$\boldsymbol{A}$	Une matrice

Nommages spécifiques

J	Une fonction de coût
σ	Une fonction d'activation
y	Une étiquette
$\tilde{y}$	Une étiquette bruitée
$\hat{y}$	Une prédiction

Opérateurs

$\boldsymbol{A} \odot \boldsymbol{B}$	Le produit d'Hadamard des matrices $\boldsymbol{A}$ et $\boldsymbol{B}$
$\frac{dy}{dx}$	La dérivée de y en fonction de x
$\frac{\partial y}{\partial x}$	La dérivée partielle de y en fonction de x
$\nabla_x y$	Le gradient de y en fonction de x
$f(x;\theta)$	Une fonction de x paramétrée par θ

Chapitre 1

Introduction

Cette thèse CIFRE a été effectuée au sein de l'entreprise Qucit [1] (pour Quantified Cities), dont le but est de créer «une meilleure expérience utilisateur pour les villes avec l'intelligence artificielle», principalement par l'optimisation des services de mobilité urbaine. Nous décrivons dans ce chapitre les problématiques de l'entreprise justifiant ces travaux de thèse, puis introduisons la structure du document permettant de présenter les contributions et leur cadre.

1.1 Contexte

L'entreprise Qucit aide les acteurs des services urbains, opérateurs privés et institutions publiques, en utilisant des méthodes d'intelligence artificielle pour améliorer la qualité de leurs services. Ces méthodes permettent d'analyser les données à disposition pour produire des prédictions (à la demande d'un service par exemple) ou d'en extraire des motifs afin de mieux les comprendre. Par exemple :

- elles permettent d'améliorer la gestion des systèmes de vélos en libre service en prévoyant la demande. Le gestionnaire peut ainsi remplir au mieux les stations ;
- les services de parkings en et hors ouvrage ainsi que leur contrôle sont améliorés. L'usage des parkings est facilité par la prédiction des disponibilités. Elles sont fournies à l'utilisateur en fonction de son heure d'arrivée. Le contrôle des paiements est optimisé par la prédiction du nombre de fraudeurs ne payant pas leur place de parking ;
- le bien-être des piétons est modélisé en fonction de l'environnement urbain, ce qui permet de guider de futurs aménagements ;
- la prédiction des incidents sur les autoroutes permet au gestionnaire d'agir au plus vite, soit en prévention pour les éviter, soit en arrivant plus vite sur les lieux s'ils n'ont pas pu être évité.

À ces fins, Qucit utilise quotidiennement des données métiers, fournies par ses clients, telles que le remplissage minute par minute des stations de vélos en libre service, les trajets effectués par les utilisateurs, le nombre de voitures dans les parkings,

1. www.qucit.com

le trafic sur les routes, etc. L'entreprise utilise également des données topographiques, récupérées sur le dépôt de données ouvertes de chaque ville ou sur OpenStreetMap [2] pour contextualiser les données métiers. Ces données topographiques permettent de dessiner des objets sur une carte : leur emplacement, leur forme, leur altitude, etc. Cette contextualisation permet d'affiner le problème traité. Par exemple : la proximité d'un bar aura un impact sur l'affluence de vélos dans une station ; la présence d'une aire d'autoroute augmente la probabilité qu'un chien traverse les voies ; une route plus large augmente le stress ressenti par les piétons.

Cependant, les erreurs présentes dans les données métiers ou dans les données topographiques altèrent la qualité des analyses. Par exemple, si on prédit que dans le quart d'heure qui suit, une station contenant 4 vélos en perdra 3, on peut penser que la station ne sera pas vide et on conseillera donc à l'utilisateur de se diriger vers celle-ci afin d'emprunter le vélo restant. Cependant, si le vélo restant n'est pas utilisable, notre conseil n'aura pas été bon. Si les données de trafic autoroutier utilisées sous estiment le trafic réel, on sous estimera le risque d'incident au risque qu'une équipe de secouristes se basant sur nos prédictions arrive trop tard sur les lieux. Pour les données topographiques, si un bar est référencé alors qu'il a fermé depuis peu, on établira des prédictions tenant compte de son influence alors qu'elle est devenue inexistante. On présentera un lieu comme agréable pour les piétons, influencé par la présence d'arbres ombrageant une place, quand ceux-ci auront été remplacés par des lampadaires. Il est donc important de pouvoir estimer si de telles anomalies sont présentes dans les données utilisées.

L'objet de cette thèse est de proposer des méthodes automatique de correction des données utilisées par l'entreprise Qucit afin d'obtenir des analyses plus justes permettant de fournir un meilleur service aux clients ainsi qu'à leurs usagers. L'amélioration des données métiers se fait au cas par cas, en fonction des méthodologies du client, de la qualité de ses données, de leur type, etc. Dans un premier temps, nous nous sommes intéressés à la détection de vélos défectueux au sein des données de trajets du système de vélo en libre service de New York. Le manque de données étiquetées nous a dirigés dans un second temps vers une autre problématique ayant un intérêt pour l'entreprise : la correction des données topographiques utilisées pour la contextualisation. À l'aide d'images satellites, nous extrayons le contour de chaque bâtiment représenté, dans le but de vérifier l'exactitude des données contenues au sein des données ouvertes d'OpenStreetMap (OSM) [Haklay et Weber, 2008].

1.1.1 Détection de vélos défectueux

Les systèmes de partage de vélos correspondent généralement à un ensemble de stations, entretenues par un opérateur au sein d'une ville, dans lesquelles les vélos peuvent être empruntés et déposés. En 2015, plus d'un million de vélos ont été partagés dans plus de 700 systèmes de vélos en libre-service dans le monde [Fishman, 2016]. Chacun de ces vélos a été fortement utilisé, voyageant jusqu'à une moyenne de dix fois par jour à Barcelone comme rapporté par Bikesharing Napoli [Napoli, 2013]. En 2017, la Chine comptait à elle seule 50 entreprises de partage de vélos, et 20 millions

2. https://www.openstreetmap.org

de vélos partagés [3]. Le nombre élevé de vélos ainsi que leur fort taux d'utilisation rendent les pannes très probables. En effet, un vélo beaucoup utilisé voit l'usure de ses pièces mécaniques augmenter, de même que la probabilité d'avoir un pneu crevé. Ainsi, de nombreux vélos sont cassés, en attente d'être réparés, immobilisés dans une station.

Le problème est tellement courant que les utilisateurs de vélos en libre-service ont pris l'habitude de tourner la selle vers l'arrière lors de la rencontre d'un vélo cassé, signalant ainsi aux autres qu'il ne vaut pas la peine d'être loué [Susi, 2013a]. Grâce à cet investissement personnel, d'autres utilisateurs évitent de perdre du temps avec la location et le retour de vélos inutilisables, mais le gestionnaire n'est pas averti de la panne. De manière générale, les gestionnaires du système sont en charge de le maintenir et effectuent des rondes : en plus de rééquilibrer les stations, ils visitent régulièrement chacune d'entre elles pour vérifier l'état de fonctionnement des vélos présents. Les vélos sont réparés sur place lorsque c'est possible (par exemple lorsque le pneu est dégonflé). Dans le cas contraire, le personnel d'entretien prend le vélo défectueux et l'apporte à un atelier spécialisé.

Dans la ville de New York, bien que le personnel d'entretien de CitiBike effectue régulièrement des tournées pour inspecter l'état de chaque vélo, un système d'alerte a été mis en place : chaque borne dispose d'un bouton qui peut être pressé afin d'alerter que le vélo est cassé. La nécessité d'un tel dispositif prouve que les rondes ne sont pas assez efficaces et que les opérateurs en sont conscients.

Dans un travail précédent cette thèse, nous avons montré que lorsqu'un système de vélos en libre-service voit son nombre de stations doubler, le nombre de trajets à vélo est triplé [Bonnotte *et al.*, 2015]. Par conséquent, le nombre de déplacements dans un système de vélos en libre-service en expansion augmente rapidement, ce qui implique que le nombre de pannes et le nombre d'utilisateurs désireux de louer un vélo augmenteront également rapidement. Cela conduit à un plus grand nombre d'utilisateurs frustrés qui se trouvent dans une situation où les seuls vélos disponibles sont cassés. Un système de détection automatique de vélos défectueux permettrait à la fois aux gestionnaires d'intervenir plus rapidement et aux utilisateurs de disposer de prédictions de disponibilités ajustées. Cette problématique fût le premier objet de nos recherches.

1.1.2 Extraction de bâtiments au sein d'images satellites

Qucit utilise des données topographiques dans tous ses outils en plus des données métier. Par exemple, dans son produit ComfortPredict [4], l'impact de la topologie d'un lieu sur le bien être des piétons est modélisé à partir de sondages géolocalisés. Pour réaliser cette modélisation, plusieurs éléments sont pris en compte : la position et la taille des arbres autour de la position étudiée, mais aussi celles des bancs, des routes, des bâtiments... Bien entendu, cette modélisation n'a de sens que si les données topologiques sont exactes.

3. http://www.chinadaily.com.cn/business/talkingbusiness/2017-08/15/content_30621103.htm
4. https://qucit.com/comfortpredict_fr/

Ces données sont récupérées depuis deux types de sources : les donnés ouvertes de la ville quand elles sont disponibles, et OSM. Les données ouvertes sont souvent précises et à jour, mais le format et le type de contenu diffère de ville en ville, rendant difficile l'automatisation de leur récupération et de leur utilisation. Au contraire, les données provenant d'OSM respectent toujours le même format et les types d'objets pouvant être rencontrés sont connus d'avance. Cependant, ces données sont collaboratives, à la manière de Wikipédia [5]. Ce sont les nombreux utilisateurs qui créent chaque élément. Les zones non cartographiées, plus à jour, ou erronées sont nombreuses et nécessitent une vérification, voir une complétion manuelle. Haklay [2010] a étudié la qualité des données Londoniennes (ville natale du projet) et anglaises. En quatre ans, 29% de la surface du pays a été cartographiée. Les objets cartographiés se trouvent à une distance moyenne de 6 mètres de la position fixée par une enquête terrain. Girres et Touya [2010] font une analyse encore plus poussée des données françaises. Ils vérifient huit critères de qualité, dont la précision géométriques, la complétude des données, la précision temporelle (mise à jour des données en fonction des changements dans le monde réel), la cohérence des données (superpositions, continuité des routes...). Ils mettent en évidence une forte hétérogénéité de la qualité des données. Ils observent la même distance moyenne aux coordonnées GPS d'un peu plus de 6 mètres. Ils montrent que la complétude des données dépend non linéairement du nombre de contributeurs actifs dans la zone cartographiée (une zone avec trois contributeurs aura 10 à 100 fois plus de contributions qu'une zone avec un contributeur). Il y a donc un réel intérêt pour une solution capable de repérer automatiquement les erreurs au sein des données de cartographie. Nous proposons dans cette thèse d'utiliser des images satellites pour corriger les données contenues dans OSM. Celles-ci sont extrêmement variées et on se limite ici à la cartographie des bâtiments pour lesquels des jeux de données ouverts existent, permettant l'organisation de concours propices à l'évolution rapide de l'état de l'art.

1.2 Contributions

Cette section présente les contributions sur ces deux problématiques développées lors de cette thèse.

1.2.1 Détection de vélos défectueux

Nous souhaitons détecter automatiquement les vélos défectueux dans les données fournies par le gestionnaire. Ainsi, il pourra les retirer au plus vite de la circulation, et Qucit pourra en tenir compte dans ses prédictions. Nous proposons un modèle pour classer l'état d'un vélo jour par jour, en décidant s'il a besoin d'être réparé ou non. Notre classification permettrait au personnel d'entretien de savoir si un vélo a besoin d'être réparé afin qu'il puisse optimiser ses rondes. De plus, il pourrait être utilisé dans les applications mobiles où les utilisateurs peuvent planifier leurs déplacements, en affichant l'icône du nombre de vélos non seulement disponibles, mais aussi utilisables.

5. https://www.wikipedia.org/

L'originalité de notre étude repose sur le fait qu'elle est centrée sur le vélo, pas sur les données de remplissage aux bornes. Cela signifie par exemple que si un vélo cassé reste immobile pendant une longue période dans une station, il sera plus pertinent d'étudier le fait que le vélo ne bouge pas plutôt que le fait que la station ne se vide pas. Afin de suivre chaque vélo, nous travaillons avec les données des trajets New-York, partagées par Motivate[6], la société exploitant le système de vélos en libre-service CitiBike[7]. Bien que la plupart des opérateurs de systèmes de vélos en libre-service ont un site Web, utilisé par les utilisateurs pour planifier leurs déplacements, qui affiche le nombre de vélos et le nombre de quais gratuits disponibles pour chaque station, peu d'opérateurs partagent leurs données de déplacements (station et heure de départ et d'arrivée des vélos identifiés) et aucun d'entre eux ne partage ses données de maintenance. Motivate expose près de trois ans de voyages [Motivate, 2016b], ainsi que des rapports mensuels indiquant, entre autres informations, le nombre de vélos réparés dans leurs ateliers Motivate [2016a]. Nous avons proposé une méthode basée sur l'apprentissage automatique pour détecter les vélos cassés.

Nous caractérisons d'abord le comportement de chaque vélo par :

— le nombre de trajets ;
— le nombre de stations distinctes visitées ;
— le nombre de trajets circulaires (même station de départ et d'arrivée) ;
— diverses statistiques sur la distance et la durée des emprunts.

Nous utilisons pour cela une fenêtre glissante temporelle de taille paramétrée (plusieurs jours), avec un pas de temps d'un jour. Une phase de partitionnement est utilisée pour regrouper différents comportements au sein de catégories. Ce regroupement permet d'extraire les comportements anormaux : nous considérons un comportement comme anormal quand il n'existe pas de groupe auquel il appartient clairement, c'est à dire que même le groupe qui lui a été attribué présente un comportement distinct du sien. À ce moment-là, nous utilisons les données des rapports mensuels comme étiquettes pour évaluer la classification. Ces données sont une agrégation des réparations de vélos sur une période d'un mois. Puisqu'il n'y a pas de correspondance un-à-un entre les exemples (un jour, un vélo) et les étiquettes (un mois, tous les vélos), nous traitons un problème d'apprentissage agrégé [Musicant *et al.*, 2007]. On vérifie que l'agrégation de nos résultats correspond à l'agrégation de la vérité terrain, mais on ne dispose pas de la vérité terrain détaillée.

En l'absence de données étiquetées plus détaillées, nous n'avons pas pu poursuivre pour améliorer ces travaux. Par souci de complétude, ce volet des travaux est néanmoins inclut dans le manuscrit, mais servira de cas d'usage pour illustrer le domaine de l'apprentissage non supervisé. Nous avons ensuite étudié un autre problème relevant de l'apprentissage automatique et s'appliquant aux problématiques de Qucit : la détection d'erreurs dans les données cartographiques.

6. https://www.motivateco.com/
7. https://www.citibikenyc.com/

1.2.2 Extraction de bâtiments au sein d'images satellites

Comme nous l'avons vu, les données métiers utilisées peuvent contenir des anomalies. Mais c'est aussi le cas des données topographiques utilisées pour la contextualisation. Pour les détecter, nous étudions la segmentation d'images satellites. Nous nous appuyons sur le jeu de données [8] du challenge SpaceNet [9] qui propose de mettre au point des modèles pour extraire automatiquement les contours des bâtiments, à partir d'images satellites. Il contient 10 593, images satellites de Paris, Vegas, Shanghai et Khartoum avec une résolution de 30cm/pixel, de taille 650×650 pixels. Cette résolution permet de distinguer clairement les bâtiments. Les images satellites étiquetées accessibles gratuitement ont généralement des résolutions bien plus faibles, entre 1.5 et 10 mètres par pixel.

Les principaux concurrents du concours SpaceNet ont soumis leur code qui est maintenant accessible au public [10]. Nous avons donc étudié la solution gagnante (un ensemble de trois modèles U-Net [Ronneberger *et al.*, 2015] fusionnés par moyenne non pondérée) afin de l'améliorer.

Nous évaluons une série de stratégies de fusion afin de trouver l'approche la mieux adaptée au problème de la détection des bâtiments. La stratégie simple existante est comparée à des stratégies plus complexes proposées : un modèle moyen pondéré, un modèle moyen pondéré avec prise en compte du voisinage, et un modèle de fusion profonde (une architecture U-Net). Toutes les solutions de fusion proposées utilisent les résultats des trois modèles U-Net précédents comme données d'entrée. Dans certains cas, l'image satellite peut également être ajoutée à l'entrée. Nous appelons ces modèles «combineurs». Nous évaluons les combineurs de complexité croissante afin de déterminer si la moyenne non pondérée ou des combineurs de faible complexité (super learners [Ju *et al.*, 2017]) sont meilleurs que les architectures CNN (combineurs profonds) pour fusionner un ensemble de segmentations.

Nous avons obtenus des résultats intéressants avec une amélioration de la métrique de performance allant jusqu'à +7%, nous menant à la 6e place du concours deepglobe [11], organisé dans le contexte d'un workshop de la conférence CVPR 2018. Les erreurs de détection rencontrées suggèrent l'utilisation de convolutions dilatées [Yu et Koltun, 2015] pour mieux détecter les petits bâtiments parmi les plus gros, ainsi que l'usage de techniques de segmentation d'instance, pour séparer les bâtiments le uns des autres. Nous espérons pouvoir mettre en place ces solutions au sein des modèles utilisés lors de nos prochaines études.

Puisque le but de cette étude est d'entraîner un modèle sur les données d'OpenStreetMap déjà disponibles, puis d'utiliser les prédictions faites pour détecter les erreurs au sein de ces mêmes données, nous quantifions l'impact des erreurs dans les données étiquetées sur la qualité des prédictions. Cette étude permet de déterminer qu'il est possible de produire des contours de bâtiments meilleurs que ceux utilisés en tant que vérité terrain lors de l'apprentissage.

8. https://aws.amazon.com/public-datasets/spacenet/
9. https://community.topcoder.com/longcontest/?module=ViewProblemStatement&rd=16835&pm=14439
10. https://github.com/SpaceNetChallenge/BuildingDetectors_Round2
11. http://deepglobe.org/

1.3 Organisation du manuscrit

Le manuscrit est organisé autour des chapitres suivants. Le Chapitre 2 permet de se familiariser avec les concepts d'apprentissage automatique, récurrents au sein de cette thèse. Y figure notamment la détection de pannes dans les systèmes de vélos en libre service qui s'apparente à la détection d'anomalies. La problématique de détection de bâtiments dans les images satellites utilise de la segmentation sémantique, ce qui fait l'objet du Chapitre 3. Le Chapitre 4 conclut cette thèse par un récapitulatif de nos travaux et une discussion sur les perspectives.

Le **Chapitre 2** permet de se familiariser avec les concepts propres à l'apprentissage automatique qu'il est nécessaire de maîtriser pour appréhender la suite du manuscrit. Les Préliminaires en **Section 2.1** permettent de comprendre ce qu'est l'apprentissage automatique et quels en sont les mécanismes principaux. Les deux types d'apprentissage automatiques utilisés sont dits «supervisé» et «non supervisé». Nous présentons d'abord ce qu'est l'apprentissage supervisé en **Section 2.2**, notamment comment on entraîne un tel modèle, quelles sont les potentielles difficultés et comment on peut les résoudre. Avant de s'intéresser à l'apprentissage non supervisé, nous abordons en **Section 2.3** les modèles d'apprentissage profond. Dans cette thèse, ils ne sont utilisés que dans un cadre d'apprentissage supervisé. Nous y expliquons ce qu'est un réseau de neurone artificiel, comment ça fonctionne et comment on l'entraîne, notamment dans le cadre du traitement d'image. Fort de cette connaissance, la **Section 2.4** présente ensuite les trois principales applications de l'apprentissage non supervisé : le partitionnement de données, la détection d'anomalies et les modèles d'apprentissage profonds générateurs. Notre contribution sur la détection de vélos défectueux au sein des systèmes de vélos en libre service permet de conclure cette section par l'illustration d'une utilisation d'apprentissage non supervisé en Section 2.5. En effet, on utilise pour résoudre ce problème du partitionnement de données au service de la détection d'anomalies.

Le Chapitre 3 traite spécifiquement du problème de détection de bâtiments au sein des images satellites. La **Section 3.1** établit un état de l'art de l'utilisation de l'apprentissage profond pour la segmentation d'images. La **Section 3.3** présente l'approche initiale que nous avons adoptée sur un jeu de données avec des images de résolution réduites par rapport à nos travaux suivants. Nous appliquons un modèle de type Segnet [Badrinarayanan *et al.*, 2015] sur différents types d'étiquettes (champ de distances, masques, multi-niveaux). La **Section 3.4** détaille notre contribution sur le thème de la fusion de modèles. On y compare la méthode de fusion la plus répandue (une moyenne non pondérée) à trois méthodes de plus en plus complexes (moyenne pondérée, moyenne pondérée avec voisinage, modèle profond). La **Section 3.5** présente une seconde étude. On y analyse l'évolution des performances d'un modèle au fur et à mesure de l'introduction de bruit dans les étiquettes. Le but est d'estimer si des zones peu cartographiées peuvent être améliorées par un modèle ayant été entraîné avec des données incomplètes ; les bâtiments non cartographiés doivent être détectés par le modèle.

En conclusion de ce manuscrit, le **Chapitre 4** récapitule en **Section 4.1** les problématiques de Qucit et comment nos travaux permettent d'y répondre et explicite les contributions apportées durant ces trois années. Il présente ensuite en

Section 4.2 les perspectives d'évolutions des travaux, en abordant notamment les points qui permettraient d'obtenir un modèle plus performant, la mise en production des modèles au sein d'un produit abouti, et l'application des techniques et méthodes acquises pendant cette thèse à d'autres problématiques.

Chapitre 2

Apprentissage automatique

Ce chapitre permet de se familiariser avec les différentes notions liées à l'apprentissage automatique, sous ses formes supervisées et non supervisées, utilisé pour résoudre les problématiques de Qucit. L'ensemble des concepts abordés peut être approfondi par la lecture de Goodfellow *et al.* [2016], dont nous reprenons ici les notations.

Le problème de détection des vélos défectueux dans les systèmes de vélos en libre service, utilisant l'apprentissage non supervisé et ayant fait l'objet d'une publication dans une conférence internationale est présenté dans la Section 2.4.

Nous appréhendons dans une première section le concept d'**apprentissage automatique**, ses mécanismes et ses enjeux. Il est important de maîtriser ces principes pour pouvoir s'intéresser dans une deuxième section à l'**apprentissage supervisé** qui est une forme d'apprentissage automatique utilisée lorsqu'on dispose d'un jeu de données permettant d'entraîner le modèle en ayant à disposition une vérité terrain, correspondant aux résultats attendus. Nous traitons dans une troisième section d'**apprentissage profond**, type de modèle central dans cette thèse, utilisé pour la détection de bâtiments au sein d'images satellites. L'**apprentissage non supervisé**, qui a fait l'objet d'une contribution au sujet de la détection de vélos défectueux au sein d'un système de vélos en libre service est présenté dans la dernière section du chapitre.

2.1 Prérequis : mécanismes de l'apprentissage automatique

L'apprentissage automatique permet à des machines de résoudre des problèmes sans avoir été explicitement programmées pour [Samuel, 1959]. On s'en sert généralement lorsque programmer explicitement un algorithme est trop complexe, souvent à cause du nombre trop élevé de règles ou de situations à décrire. Un système d'apprentissage automatique va apprendre des conséquences de ses actions afin d'optimiser ses actions ou ses réponses en fonction des données qu'il a pu analyser. Les problèmes nécessitant des techniques d'apprentissage automatique pour être résolus sont parfois triviaux pour un être humain, sans pour autant que l'on sache traduire nos solutions

en algorithmes. Décrypter les émotions d'une personne à travers les expressions de son visage est par exemple quelque chose que nous faisons instinctivement, mais il nous est pourtant impossible de décrire quantitativement toutes les propriétés du visage nous permettant de parvenir à une conclusion. On pourra avoir recourt à des modèles de reconnaissance d'image qui apprendront à résoudre ce problème en apprenant sur de nombreux exemples [Zeng *et al.*, 2018; Yu et Zhang, 2015].

L'apprentissage automatique peut être décomposé en trois principales catégories : l'apprentissage supervisé, non supervisé et par renforcement. Nous ne traiterons pas ici de l'apprentissage par renforcement puisqu'aucune démarche durant cette thèse n'en a nécessité l'usage.

Cette section décrit de quelle manière un système d'apprentissage automatique peut apprendre à résoudre un problème. D'abord en décrivant comment un problème est modélisé et de quelle manière une fonction de coût est utilisée pour quantifier l'atteinte de l'objectif, puis comment, par l'optimisation de ses paramètres, un modèle d'apprentissage peut évoluer pour atteindre cet objectif.

2.1.1 Modélisation d'un problème d'apprentissage automatique

Un programme d'apprentissage automatique est un modèle statistique apprenant de ses expériences afin de de maximiser la probabilité d'obtenir le résultat attendu. Selon Goodfellow *et al.* [2016], la nature des expériences diffère selon que le modèles relève de l'apprentissage supervisé ou non supervisé. Pour un modèle non supervisé, expérimenter consiste à observer plusieurs vecteurs aléatoires $\mathbf{x}$ du jeux de données analysé pour en apprendre la distribution $p(\mathbf{x})$, ou certaines propriétés de cette distribution. Pour un modèle supervisé, expérimenter consiste à observer plusieurs vecteurs aléatoires $\mathbf{x}$ et une valeur ou un vecteur associé $\mathbf{y}$ afin d'apprendre à prédire $\mathbf{y}$ en fonction de $\mathbf{x}$, habituellement en estimant $p(\mathbf{y}|\mathbf{x})$. On appelle $\mathbf{y}$ une étiquette, ou la vérité terrain.

Formaliser un problème d'apprentissage automatique revient, selon Goodfellow *et al.* [2016], à décrire comment le modèle doit traiter un exemple. Un exemple est représenté ici par ses propriétés, sous forme de tenseur (souvent d'ordre 1, un vecteur). On traite typiquement un vecteur $\mathbf{x} \in \mathbb{R}^n$ pour un exemple représenté par n propriétés différentes. Dans le cas de la segmentation d'images, traité dans le Chapitre 3.1, un exemple est une image (tenseur d'ordre 3 : hauteur, largeur et profondeur) et ses propriétés correspondent aux valeur des pixels contenus dans l'image.

Le résultat produit en sortie du modèle dépend du problème posé. Ce peut être une classification binaire (ce patient est il atteint d'un cancer ou non, ce mail est-il indésirable) ou non binaire (cette photo est-elle une photo de chat, de chien, oiseau, etc), une régression qui est une prédiction d'un phénomène continu (estimation du prix d'un bien, de l'âge d'un objet, etc), une traduction d'une représentation à une autre (typiquement une traduction d'un texte depuis une langue vers une autre), etc. Cette liste n'est pas exhaustive.

Pour résoudre un problème d'apprentissage automatique il faut d'abord choisir

le type de modèle à entraîner. On présente ci-après quelques exemples de modèles qu'il est possible d'utiliser.

2.1.1.1 Exemples de modèles supervisés

L'exemple le plus simple de modèle supervisé est le modèle linéaire. Celui-ci calcule une combinaison linéaire des propriétés de l'exemple en entrée dont les poids sont les paramètres du modèle. L'équation du modèle est la suivante :

$$f(\boldsymbol{x}; \boldsymbol{w}) = \boldsymbol{x}^T * \boldsymbol{w} \tag{2.1}$$

où f est le modèle qui applique au vecteur de propriétés $\boldsymbol{x}$ une combinaison linéaire paramétrée par le vecteur $\boldsymbol{w}$.

Une régression logistique est une adaptation du modèle linéaire pour la classification. On applique une fonction logistique pour restreindre l'image du modèle à l'intervalle $[0; 1]$:

$$f(\boldsymbol{x}; \boldsymbol{w}) = \frac{1}{1 + e^{-\boldsymbol{x}^T * \boldsymbol{w}}} \tag{2.2}$$

Les problèmes de classification sont également traités à l'aide de Machines à Vecteurs de Support [Hearst, 1998](SVM), qui cherchent le séparateur qui maximise la marge entre deux classes, ou d'arbres de décisions, souvent utilisés sous forme de forêts aléatoires [Breiman, 2001] ou de gradient boosting [Friedman, 2001]. Ces modèles ne sont pas abordés dans ce manuscrit.

2.1.1.2 Exemples de modèles non supervisés

Les problèmes non étiquetés peuvent aussi être traités l'apprentissage automatique. Ce type de modèle fait l'objet de la Section 2.4.

K-Means est un algorithme de partitionnement (voir Section 2.4.1) facile à comprendre et implémenter, et donc très répandu. On associe à $\boldsymbol{x}$ la classe c dont la moyenne des éléments $\bar{\boldsymbol{x}}_c$ est la plus proche de $\boldsymbol{x}$:

$$f(\boldsymbol{x}) = c, c \quad minimise(\bar{\boldsymbol{x}}_c - \boldsymbol{x})^2 \tag{2.3}$$

L'analyse en composante principale [Jolliffe, 2011] est un algorithme d'apprentissage non supervisé (voir Section 2.4) utilisé pour effectuer un changement de repère dans un espace dont les dimensions sont des composantes non corrélées. On s'en sert souvent pour effectuer une réduction de dimensionnalité. On citera également les réseaux de neurones artificiels, traités en Section 2.3, particulièrement efficaces pour l'analyse d'image, mais demandant de grandes quantités de données pour leur apprentissage. Cette liste n'est pas exhaustive mais montre qu'une panoplie importante de modèles est à notre disposition en fonction du type de problème à modéliser et du contexte.

2.1.1.3 Entraînement

Pour chacun de ces modèles l'entraînement, consistera à trouver les paramètres lui permettant de résoudre au mieux le problème voulu. Les paramètres différencient deux modèles du même type ayant été entraîné avec des données différentes. L'hypothèse, faite dans le choix du modèle, est adaptée au problème traité grâce aux paramètres. L'optimisation des paramètres est traitée en Section 2.1.3.

Les hyper-paramètres d'un modèle sont des paramètres qui ne sont pas optimisés lors de l'apprentissage, mais par l'utilisateur du modèle. Ils servent généralement à guider l'apprentissage pour réguler le sur-apprentissage (traité en Section 2.2.1). Dans le cas d'une régression linéaire de type LASSO, un hyper-paramètre α sera responsable de pondérer la régularisation (traitée en Section 2.2.2), amenant à limiter le nombre de termes utilisés dans la combinaison linéaire. Dans l'algorithme K-Means, K le nombre de partitions est un hyper-paramètre, on le choisit avant de trouver les centres optimaux. Pour fixer les hyper-paramètres on peut utiliser des algorithmes de recherche tels que la recherche en grille (recherche exhaustive bornée), l'optimisation Bayesienne [Močkus, 1975], des algorithmes génétiques [Di Francescomarino *et al.*, 2018], etc. Il est également possible de se baser sur les conseils d'experts dans le domaine, ou sur la similitude du problème traité à d'autres problèmes pour lesquels les hyper-paramètres optimaux sont connus. Cette optimisation est traitée en Section 2.2.3.

2.1.2 Fonction de coût

La fonction de coût est la mesure que l'apprentissage du modèle va optimiser. Les paramètres optimaux sont ceux permettant de minimiser la fonction de coût. Il faut donc veiller à ce qu'une optimisation de cette mesure corresponde à une optimisation de la tâche à effectuer par le modèle.

Dans le cas de la régression linéaire, on cherche typiquement à minimiser la norme L^2 suivante :

$$\min_{\boldsymbol{w}} J(\boldsymbol{w}; \boldsymbol{X}) = ||\boldsymbol{X} * \boldsymbol{w} - \boldsymbol{y}||_2^2 \tag{2.4}$$

où $\boldsymbol{y}$ est le vecteur contenant le résultat attendu pour chacun des exemples, $\boldsymbol{X}$ est la matrice contenant les propriétés pour chaque exemple, et $\boldsymbol{w}$ le vecteur contenant les paramètres du modèle. Malgré leur simplicité, les modèles de ce type sont utilisés quand un grand nombre de propriétés est extrait des exemples, du fait de leur propension à ne pas facilement sur-apprendre (voir Section 2.2.1), grâce aux régularisations (voir Section 2.2.2) suivantes :

Une régularisation L^2 transforme ce modèle en une régression de Ridge [Hoerl et Kennard, 2000] :

$$\min_{\boldsymbol{w}} J(\boldsymbol{w}; \boldsymbol{X}) = ||\boldsymbol{X} * \boldsymbol{w} - \boldsymbol{y}||_2^2 + \alpha {||\boldsymbol{w}||_2}^2 \tag{2.5}$$

et une régularisation L^1 en une régression LASSO [Tibshirani, 1996] :

$$\min_{\boldsymbol{w}} J(\boldsymbol{w}; \boldsymbol{X}) = ||\boldsymbol{X} * \boldsymbol{w} - \boldsymbol{y}||_2^2 + \alpha ||\boldsymbol{w}||_1 \tag{2.6}$$

α est dans les deux cas un hyper-paramètre, un paramètre qui ne sera pas appris par le modèle. Cette notion est détaillée en Section 2.2.3.

Pour chaque type de résultat attendu, il peut exister plusieurs manières d'en mesurer la qualité. Dans le cas d'une classification binaire, la sortie du modèle sera $\hat{y} \in \{0; 1\}$, à comparer avec $y \in \{0; 1\}$ la vérité terrain.

La régression logistique étant une adaptation de la régression linéaire pour répondre à ce problème, on cherche à minimiser :

$$\min_{w,c} J(\boldsymbol{w}; \boldsymbol{X}) = \sum_{i=1}^{n} \log\left(\exp\left(-y_i\left(\boldsymbol{x}_i^T \boldsymbol{w}\right)\right) + 1\right) \tag{2.7}$$

où i est le $i$ème exemple parmi n. Les mêmes régularisations que pour les modèles linéaires peuvent être ajoutées.

En utilisant d'autres types de modèles de classification, on va par exemple utiliser la précision (parmi les cas positifs trouvés, quelle est la proportion de cas réellement positifs) et le rappel (parmi les cas réellement positifs, quelle proportion a été trouvée). Plus formellement :

$$precision = \frac{TP}{TP + FP}, rappel = \frac{TP}{TP + FN} \tag{2.8}$$

où parmi T exemples

$$TP = \sum_{t=1}^{T} \hat{y}_t * y_t \tag{2.9}$$

est le nombre de vrais positifs, les cas réellement positifs prédits positifs,

$$FP = \sum_{t=1}^{T} \hat{y}_t * (1 - y_t) \tag{2.10}$$

est le nombre de faux positifs, les cas réellement négatifs prédits positifs, et

$$FN = \sum_{t=1}^{T} (1 - \hat{y}_t) * y_t \tag{2.11}$$

est le nombre de faux négatifs, les cas réellement positifs prédits négatifs.

Les mesures d'erreur couramment employées pour ces problèmes sont généralement, mais pas exclusivement, basées sur ces métriques. C'est le cas des indices de Dice [Dice, 1945] et de Jaccard [Jaccard, 1901]. Ils permettent d'obtenir une mesure unique dépendant à la fois de la précision de la prédiction, et du rappel.

$$Dice = \frac{2TP}{2TP + FP + FN} \tag{2.12}$$

$$Jaccard = \frac{TP}{TP + FP + FN} \tag{2.13}$$

On trouve régulièrement d'autres fonctions de coûts utilisées pour la classification, telles que l'entropie croisée, la fonction logistique, ou encore la fonction de

Hinge [Rosasco *et al.*, 2004], régulièrement utilisée pour l'entraînement des SVM. Ces fonctions n'ont pas été utilisées dans les travaux présentés ici, nous ne les détaillerons donc pas plus en avant.

Dans le cas d'une classification non binaire, il est possible d'adapter les mesures de précision et de rappel, par exemple en faisant une moyenne sur le nombre de classes. Pour notre problème de segmentation d'images, nous verrons que l'indice de Jaccard a été utilisé, en considérant la classification de chaque pixel individuellement dans le calcul de la précision et du rappel.

Pour les cas de régression où $\hat{y}$ et y sont des réels définis dans un ensemble possiblement borné, on utilise souvent l'erreur quadratique moyenne (RMSE, de l'anglais *Root Mean Square Error*), $\sqrt{\frac{\sum_{t=1}^{T}(\hat{y}_t - y_t)^2}{T}}$. Encore une fois, de nombreuses autres fonctions de coûts peuvent être utilisées. Aucun problème de régression n'est traité au sein de cette thèse. On utilisera cependant la RMSE afin d'illustrer nos propos. L'optimisation de celle-ci va permettre au modèle de prédire un nombre aussi proche que possible du nombre attendu, c'est la raison pour laquelle cette mesure d'erreur est adéquate au problème de régression. Elle ne serait pas adéquate pour un problème de classification par exemple, car il n'est pas forcément possible de définir une distance entre la classe prédite et la classe attendue.

Certains algorithmes de partitionnement de données ont également recours à la minimisation d'une distance. C'est le cas notamment du partitionnement K-Means [MacQueen, 1967]. Il est défini ainsi : étant donné un ensemble d'observations $\boldsymbol{X} = \boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_T}$ où chaque observation est un vecteur multidimensionnel de réels, il s'agit de regrouper les T observations en K partitions $S = S_1, S_2, ..., S_K$ en minimisant la somme des distances intra-partition. La fonction de coût est formellement définie dans l'équation 2.14 où μ_k est la moyenne des points appartenant à la partition S_k.

$$f_k(\boldsymbol{X}, S) = \sum_{i=k}^{K} \sum_{\boldsymbol{x} \in \boldsymbol{S_k}} ||\boldsymbol{x} - \boldsymbol{\mu_k}||^2 \tag{2.14}$$

2.1.3 Optimisation des paramètres

L'apprentissage d'un modèle signifie que ses paramètres sont optimisés de manière à minimiser la fonction de coût. Les méthodes d'optimisation dépendent principalement du choix du modèle.

Des solutions analytiques peuvent être employés pour certains modèles. La régression linéaire admet dans certains cas une solution analytique. Si $\boldsymbol{X}$ la matrice dont chaque ligne est un exemple $\boldsymbol{x}_i$ et chaque colonne une propriété est plus haute que large (il y a plus d'exemples que de propriétés) et que ses colonnes sont linéairement indépendantes (aucune colonne n'est le résultat de la combinaison linéaire des autres colonnes), alors on peut trouver le vecteur de paramètres optimaux $\boldsymbol{w}$ grâce à l'équation suivante :

$$\boldsymbol{w} = (\boldsymbol{X}^T * \boldsymbol{X})^{-1} * \boldsymbol{X}^T * \boldsymbol{y} \tag{2.15}$$

En pratique, des factorisations sont utilisées pour éviter d'avoir à inverser la matrice $\boldsymbol{X}^T * \boldsymbol{X}$. En effet, pour des matrices de grandes dimensions cette inversion est très coûteuse en temps de calcul. Quand on ne peut pas garantir la non colinéarité des colonnes de $\boldsymbol{X}$, on utilise généralement la descente du gradient.

La descente du gradient [Cauchy, 1847] est la principale technique utilisée dans le cas d'un modèle basé sur des combinaisons linéaires, tel que LASSO ou un modèle d'apprentissage profond. Le principe est le suivant. Soit une fonction de coût $J(w) = y, (w, y) \in \mathbb{R}^2$ et sa dérivée $J'(w)$. Cette dérivée donne la pente de la courbe de J au point d'abscisse w. Cette information permet d'estimer l'évolution de la valeur de $J(w)$ lorsqu'on change la valeur de w d'une quantité infinitésimale ϵ : $J(w + \epsilon) \approx J(w) + \epsilon * J'(w + \epsilon)$.

Afin de trouver le minimum de la fonction, on fait donc croître w si $J'(w) < 0$, décroître w si $J'(w) > 0$. Dans le cas où $J'(w) = 0$, on ne sait pas dans quel sens modifier w pour continuer à minimiser la fonction, c'est un point problématique. Lorsque $J'(w) = 0$ on a soit atteint un minimum local, un w pour lequel $J(w)$ est inférieur aux autres valeurs de J pour tous les w' aux alentours, soit atteint un point où la fonction est plate, parce qu'elle est décroissante sans être strictement décroissant.

Dans le cas général, $\boldsymbol{w}$ est un vecteur de paramètres. Le gradient est le vecteur contenant toutes les dérivées partielles, noté $\nabla_{\boldsymbol{w}} J(\boldsymbol{w})$. L'algorithme de descente du gradient est alors :

- à $t = 0$, le premier pas, $\boldsymbol{w}$ est initialisé aléatoirement au point de l'espace $\boldsymbol{w}_{t_0}$.
- calculer le gradient $\nabla_{\boldsymbol{w}} J(\boldsymbol{w})$ sur l'ensemble d'apprentissage à cet endroit de l'espace des paramètres
- Pour obtenir la valeur de $\boldsymbol{w}$ aux pas suivants, descendre le gradient : $\boldsymbol{w}_{t_{n+1}} = \boldsymbol{w}_{t_n} - \gamma \nabla_{\boldsymbol{w}} J(\boldsymbol{w})$.
- Répéter jusqu'à ce que le gradient soit aussi proche du vecteur nul que souhaité.

On dit que l'apprentissage a convergé quand $\boldsymbol{w}_{t_n} - \boldsymbol{w}_{t_{n+1}}$ est quasi nul. Une descente de gradient est illustrée en Figure 2.1.

γ est le taux d'apprentissage : plus ce dernier est élevé plus les pas faits en direction du minimum sont grands. Si γ est trop petit, l'apprentissage mettra du temps à converger. Si γ est trop grand, l'apprentissage peut ne pas converger du tout, car la descente du gradient se faisant à trop grand pas, on risque de ne jamais atteindre le point de convergence. Pire même, on risque de diverger, l'erreur croîtrait alors au fur et à mesure de l'apprentissage.

Le problème de cette approche est qu'il est nécessaire de calculer le gradient sur l'entièreté de l'ensemble d'apprentissage à chaque étape, ce qui peut être long. Pour palier à cela, la descente du gradient stochastique [Kiefer *et al.*, 1952] propose de mettre à jour les paramètres pour chaque exemple de l'ensemble d'apprentissage : $\boldsymbol{w}_{t_{n+1}} = \boldsymbol{w}_{t_n} - \gamma \nabla_{\boldsymbol{w}} J(\boldsymbol{w}; \boldsymbol{x_i}, y_i)$. Cette opération permet de passer plus facilement de minimums locaux en minimums locaux, au risque de ne pas converger vers un minimum global. Il faudra réduire progressivement le taux d'apprentissage pour converger.

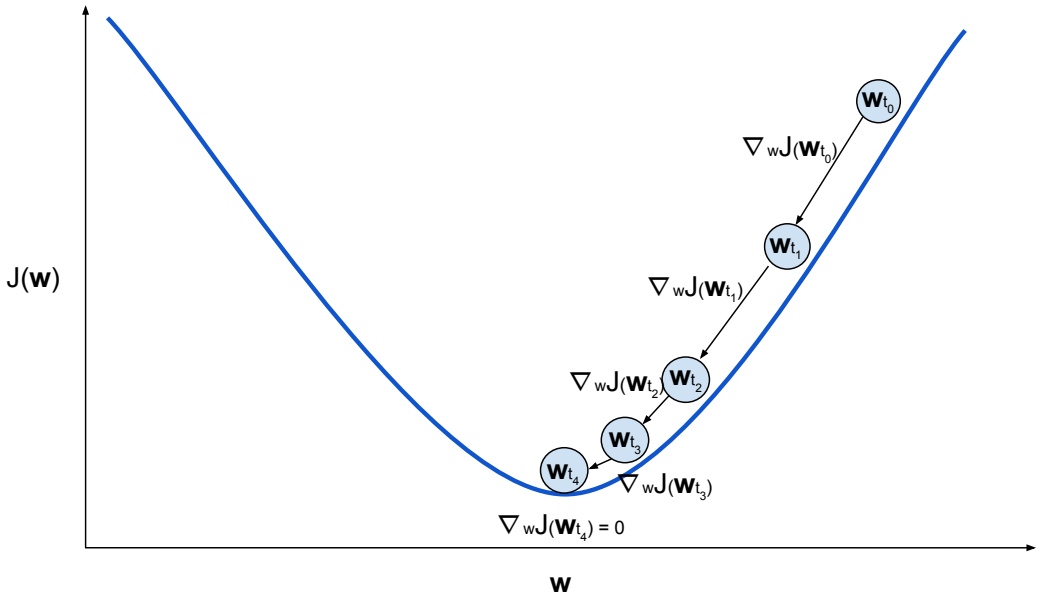

FIGURE 2.1 – Illustration d'une descente de gradient à 4 étapes. Les poids du modèle sont initialisés à un endroit aléatoire de l'espace des paramètres, $\boldsymbol{W}_{t_0}$. Le gradient $\nabla_{\boldsymbol{w}} J(\boldsymbol{W}_{t_0})$ est utilisé pour calculer la mise à jour des poids. Le processus est répété à chaque nouvelle position jusqu'à ce que le gradient soit nul : un minimum local a été atteint.

La descente du gradient par mini-batch est un entre deux : on calcule le gradient sur des sous parties de l'ensemble d'apprentissage, typiquement entre 50 et 256 exemples. La descente du gradient stochastique est un cas particulier de la descente par mini-batch, pour laquelle la taille du mini-batch est de 1.

Pour optimiser le temps de convergence, la notion de momentum est introduite. On conserve l'élan des précédentes itérations pour traverser plus facilement des zones où le gradient est faible. On obtient alors un vecteur de mise à jour $\boldsymbol{v_{t_n}} = \eta \boldsymbol{v_{t_{n-1}}} + \gamma \nabla_{\boldsymbol{w}} J(\boldsymbol{w})$, utilisé pour mettre à jour les paramètres : $\boldsymbol{w}_{t_{n+1}} = \boldsymbol{w}_{t_n} - \boldsymbol{v_{t_n}}$.

L'accélération de Nesterov [Nesterov, 1983] est une amélioration du momentum qui prend en compte le fait que l'on connaisse d'avance $\boldsymbol{v_{t_{n-1}}}$. Plutôt que de calculer le gradient à l'endroit de l'espace où on se trouve actuellement, il propose de le calculer à l'endroit où l'on se trouvera après avoir utilisé le momentum : $\boldsymbol{v_{t_n}} = \eta \boldsymbol{v_{t_{n-1}}} + \gamma \nabla_{\boldsymbol{w}} J(\boldsymbol{w} - \boldsymbol{v_{t_{n-1}}})$. Cette différence est illustrée en Figure 2.2.

L'utilisation de cet algorithme suggère que la fonction de coût soit continue et dérivable sur son ensemble de définition.

D'autres algorithmes sont utilisés lorsque la fonction de coût n'est pas dérivable. Dans le cas du problème K-Means, l'algorithme le plus répandu [Lloyd, 1982] est itératif. Le centre de chaque partition est initialisé de manière aléatoire. L'algorithme alterne entre deux phases. À chaque élément x_t est attribuée la partition dont le centre est le plus proche de lui. Quand tous les éléments se sont vus attribuer une

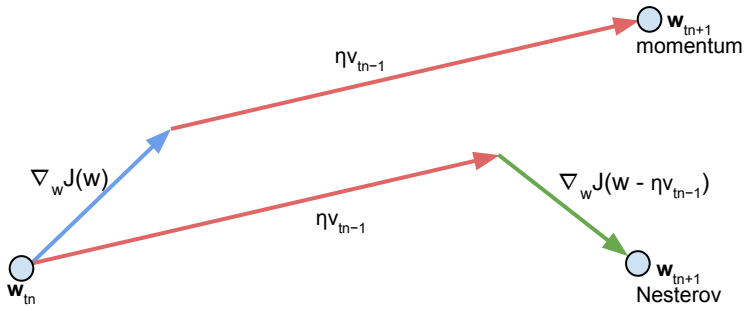

FIGURE 2.2 – Les méthodes de momentum et momentum accéléré par Nesterov n'arrivent pas au même résultat. Le gradient calculé au point de départ (bleu) est en effet différent de celui (vert) calculé après la translation par le vecteur du momentum (rouge). L'accélération de Nesterov permet de calculer le gradient en tenant compte du fait que la position va être modifiée par le momentum.

partition, les centres des partitions sont déplacés à la moyenne de leurs éléments. Le processus est répété jusqu'à stabilisation. Il n'est pas garanti de converger vers la solution optimale.

On citera également la résolution de contraintes utilisée pour optimiser les SVM avec l'utilisation notamment de multiplicateurs de Lagrange [Joachims, 1998].

Les méta heuristiques sont un autre exemple de technique d'optimisation. Ce sont des stratégies guidant le processus de recherche de paramètres quasi-optimaux. Elles ne sont pas spécifiques à un problème et ne sont généralement pas déterministes. Le recuit simulé [Kirkpatrick *et al.*, 1983] est un exemple de méta heuristique réputée, utilisée pour trouver les extrema d'une fonction.

Enfin, les algorithmes génétiques sont des processus d'optimisations inspirés de la théorie de l'évolution de Darwin. En utilisant une fonction capable d'attribuer un score (à maximiser) à chaque individu, on procède à une «sélection naturelle». Chaque individu est caractérisé par ses «gènes» (propriétés), initialisés aléatoirement pour la première génération. On sélectionne en fonction de leur score les individus qui passeront leurs gènes à la génération suivante (combinaison des propriétés des parents), et on applique avec une faible probabilité des «mutations» (altérations aléatoires de propriétés). Les individus s'améliorent au fur et à mesure des générations.

Que l'apprentissage soit supervisé ou non, l'apprentissage passe donc par une optimisation des paramètres internes du modèle. Nous voyons dans la section suivante quels sont les enjeux plus particuliers de l'apprentissage supervisé et quelles problématiques l'optimisation des paramètres soulève dans ce contexte.

2.2 Apprentissage supervisé

Les données utilisées pour l'apprentissage supervisé sont *étiquetées* : on connaît le résultat auquel le modèle doit parvenir pour chaque exemple utilisé lors de l'apprentissage. Le but du modèle est d'apprendre à associer chaque exemple à une étiquette

correcte.

L'apprentissage supervisé est défini par Cunningham *et al.* [2008] en ces termes : «L'apprentissage supervisé implique l'apprentissage d'une mise en correspondance entre un ensemble de variables d'entrée $\mathcal{X}$ et une variable de sortie $\mathcal{Y}$ et en appliquant cette correspondance pour prédire les sorties pour des données non visualisées».

Le but est donc de déduire une fonction $f\colon \mathcal{X} \to \mathcal{Y}$ d'un ensemble de d'apprentissage $\mathcal{A}_n$ composé de paires (exemple, label) tel que

$$\mathcal{A}_T = ((\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_T, y_T)) \in (\mathcal{X} \times \mathcal{Y})^T$$

où $\boldsymbol{x}_t, t \in 0, ..., T$ est généralement défini dans $\mathbb{R}^d$ et y_t est généralement défini soit dans $\mathbb{R}$ pour un problème de régression, soit dans une sous partie de $\mathbb{N}$ pour un problème de classification. On a donc un problème de régression quand il s'agit de prédire une valeur numérique continue (par exemple prédire le prix d'une maison) et un problème de classification quand il s'agit d'assigner une classe (par exemple déterminer si une image est celle d'un chien ou d'un chat).

Dans le cas de l'apprentissage supervisé, la fonction de coût est utilisée pour mesurer la différence entre ce qui a été produit par le modèle $f(\boldsymbol{x}) = \hat{y}$ et ce qui était attendu, le label y.

L'apprentissage d'un modèle supervisé étant un compromis entre les états de sous et de sur apprentissage, nous étudions ces notions en Section 2.2.1. Nous verrons en Section 2.2.2 comment la régularisation permet d'accroître le contrôle de ces états, et finalement en Section 2.2.3 que les hyper-paramètres permettent d'optimiser le processus d'apprentissage et particulièrement la régularisation.

2.2.1 Sous-apprentissage et sur-apprentissage

Lorsqu'on utilise un modèle d'apprentissage automatique en production, on ne dispose plus d'étiquettes. On doit pouvoir se fier aux prédictions faites par le modèle. Pour cela, on distingue la phase d'apprentissage du modèle, celle de test et celle de production. On sépare le jeu de données étiquetées en trois parties : les ensembles d'apprentissage, de validation, et de test. Lors de la phase d'apprentissage le modèle est entraîné sur l'ensemble d'apprentissage. Ensuite, on utilise l'ensemble de test pour mesurer l'erreur du modèle sur des données qu'il n'a jamais vues. On espère ainsi pouvoir juger des performances du modèle une fois en production. L'ensemble de validation sert à contrôler l'apprentissage du modèle, particulièrement le choix des hyper-paramètres. Cette séparation n'est utile que sous la condition que chaque ensemble est indépendant des autres et observe la même distribution.

Le biais est l'a-priori que le modèle a sur les données, il est source d'erreur. Sans apprentissage, le biais est très fort. Il décroît au fur et à mesure de l'apprentissage, quand le modèle apprend la distribution des données. La variance est la sensibilité du modèle à de petites variations au sein des données. Au début de l'apprentissage la variance est très faible, puis croît au fur et à mesure que le modèle s'adapte aux données. L'apprentissage faisant décroître le biais et croître la variance, le but est de trouver le compromis qui permettra de minimiser l'erreur. Une part irréductible de l'erreur est due au bruit présent dans les données d'apprentissage.

Lorsque la fonction de coût est minimisée lors de la phase d'apprentissage, on s'attend à ce qu'elle soit également faible sur de nouvelles données, quand le modèle entraîné entre en production. Le sur-apprentissage apparaît lorsque l'erreur du modèle sur les données de test est significativement plus élevée que sur les données d'apprentissage. Le sous-apprentissage correspond à une erreur élevée sur l'ensemble d'apprentissage Ces cas de figure sont illustrés en Figure 2.3.

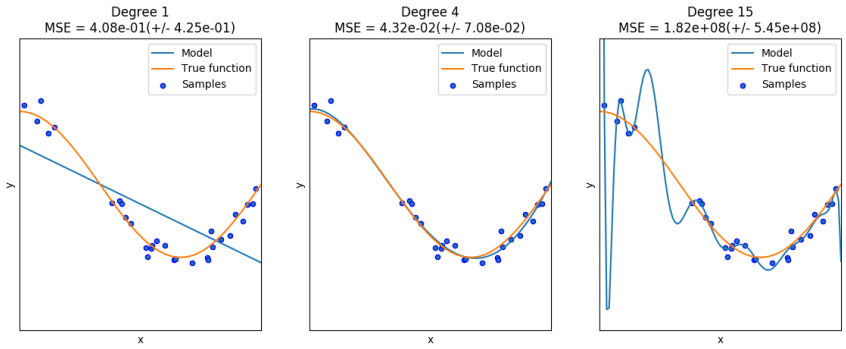

FIGURE 2.3 – Image tirée de la documentation de scikit-learn [Pedregosa *et al.*, 2011] [1]. On représente en rouge la fonction réellement décrites par les données. En bleu on retrouve les données d'entraînement. La ligne verte est la fonction décrite par le modèle de manière à minimiser l'erreur faite sur les données d'entraînement.
Le premier cas illustre un sous-apprentissage. Le modèle n'est pas assez complexe. Ne pouvant décrire que des droites, il ne peut pas modéliser la fonction voulue. Le biais est haut, la variance basse.
Le deuxième cas illustre un apprentissage correcte. Avec le bon niveau de complexité, le modèle a su approcher la fonction voulue de très près. Le biais et la variance ne peuvent décroître sans observer une croissance de l'autre.
Le troisième cas illustre un sur-apprentissage. Le modèle trop complexe a su passer par la plupart des points appartenant aux données d'entraînement. On voit qu'à certains endroits la courbe du modèle (en bleu) est très éloignée de la courbe décrivant la véritable fonction (en rouge). Si dans le jeu de test des données se trouvent à cet endroit, l'erreur sera très grande. Le biais est faible mais la variance haute.

Ces situations sont souvent liées à la complexité du modèle. Un modèle raisonnablement complexe a une plus grande latitude pour décrire les événements qu'il doit modéliser. Il va être capable d'apprendre plus de types de fonctions. Si le modèle n'est pas assez complexe, il ne sera pas capable de décrire la fonction qu'il est censé modéliser et l'erreur sera haute. S'il est trop complexe il pourra modéliser des fonctions qui collent parfaitement aux données d'apprentissage malgré le bruit dans

1. http://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html

ces données, il sera trop sensible aux petites variations présentent dans les données. L'ensemble des fonctions qu'un modèle est capable de modéliser est appelé l'espace des hypothèses. La Figure 2.4 montre l'évolution de l'erreur faite sur les données d'apprentissage et celles de test en fonction de la complexité du modèle. La complexité optimale d'un modèle doit approcher la complexité réelle du problème traité, mais elle n'est pas connue d'avance.

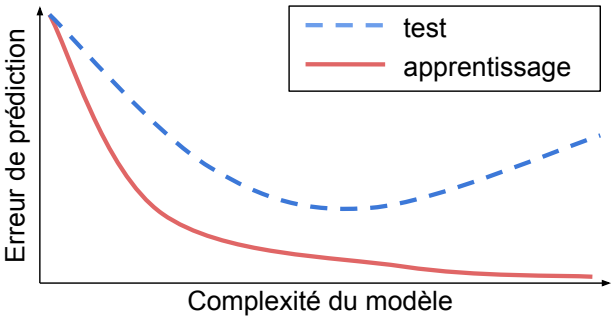

FIGURE 2.4 – Alors que la complexité du modèle augmente, l'erreur faite sur les données d'apprentissage ne fait que diminuer. En effet, le modèle a de plus en plus de moyens pour les décrire correctement. Alors que le modèle s'améliore, l'erreur sur les données de test va elle aussi diminuer. Quand ce n'est plus le cas, et qu'augmenter la complexité du modèle augmente l'erreur sur les données de test, c'est que l'on est entrain de sur-apprendre.

Reprenons l'exemple de la RMSE, $\sqrt{\frac{\sum_{t=1}^{T}(\hat{y}_t - y_t)^2}{T}}$ où T est le nombre d'exemples dans les données d'apprentissage, et d'un modèle linéaire défini ainsi : $f(\mathbf{x}) = \sum_{n=0}^{c} \theta_n x_n^n$ où c est l'hyper-paramètre qui représente le degré du polynôme et les θ sont les paramètres du modèle. Le modèle est linéaire car c'est une combinaison linéaire de ses paramètres, même si le vecteur de propriétés est composé de transformation non linéaires de x_0. Si $c >= T$ alors l'erreur sur les données d'apprentissage sera nulle, car le modèle sera capable de décrire une courbe passant par chacun des points du jeu de données.

Cet hyper-paramètre permet de contrôler l'espace des hypothèses en définissant quel type de fonction peut être utilisé comme hypothèse, ici en choisissant le degré maximum du polygone.

2.2.2 Régularisation

La complexité du modèle peut être contrôlée au delà de l'espace des hypothèses. On peut faire en sorte que parmi deux fonctions présentent dans cet espace et produisant les mêmes résultats, le modèle préfère la solution la moins complexe grâce à la régularisation. Elle se présente en général sous la forme d'un terme additionné à la fonction de coût initiale. Ce terme croît avec la complexité du modèle, faisant ainsi croître la fonction de coût à minimiser. Cependant, augmenter la complexité du modèle permet de mieux approximer les données, et ainsi réduire la fonction de

coût, c'est le compromis entre biais et variance que l'on a vu précédemment, qui est trouvé dans le cas de la régularisation grâce à une pondération.

En se basant une fois encore sur la RMSE, on peut ajouter à cette erreur un terme de *régularisation* permettant de pénaliser les paramètres trop élevés qui donneraient donc trop d'importance à un des éléments du vecteur $\boldsymbol{x}$: $\sum_{n=0}^{c} \theta_n^2$. La fonction de coût devient alors : $\sqrt{\frac{\sum_{t=1}^{T}(\hat{y}_t - y_t)^2}{T}} + \lambda \sum_{n=0}^{c} \theta_n^2$. Cette régularisation est dite de norme L^2. λ est le coefficient de régularisation. C'est avec cet hyper-paramètre qu'on décide de l'importance de la régularisation. S'il est nul, seule l'erreur quadratique moyenne est utilisée, et plus il est élevé moins cette dernière sera importante face à la pénalisation du poids des paramètres. Puisque λ est un hyper-paramètre du modèle, il ne sera pas optimisé durant l'apprentissage. En effet, la descente du gradient ne modifiera pas λ, mais seulement θ. L'obtention d'un modèle ayant une erreur suffisamment basse (cette valeur dépendant de la fonction de coût choisie et du problème traité) passe donc par une phase d'optimisation des hyper-paramètres, qui fait l'objet de la Section 2.2.3.

Les formes les plus communes de régularisation sont les régularisations de normes L^1 et L^2. La norme L^1 correspond à la somme de la valeur absolue des paramètres du réseau et la norme L^2 à la somme de leur carré. La norme L^1 permet d'annuler un grand nombre de paramètres de manière à rendre creuses les matrices de poids. Cette sélection automatique de propriétés est notamment mise à profit au sein du modèle LASSO [Tibshirani, 1996]. Elle n'admet cependant pas de solution analytique contrairement à la norme L^2. Cette dernière est donc calculée de manière plus efficace. L'effet de la norme L^2 sur les paramètres optimaux est illustré en Figure 2.5.

2.2.3 Optimisation des hyper-paramètres

Comme nous l'avons vu précédemment, l'apprentissage optimise les paramètres d'un modèle, mais pas ses hyper-paramètres. En effet, ces derniers sont utilisés pour contrôler le processus d'apprentissage. Ils sont mis à jour en dehors du processus d'apprentissage. Cette section présente donc comment les hyper-paramètres sont optimisés.

2.2.3.1 Validation

On a vu en Section 2.2.1 que le jeu de données était séparé en trois ensembles, apprentissage pour entraîner le modèle, test pour l'évaluer et validation pour choisir les hyper-paramètres. Ça n'aurait pas de sens d'utiliser l'ensemble d'apprentissage pour cela, puisque l'hyper-paramètre choisi serait toujours celui minimisant l'erreur sur cet ensemble, conduisant inévitablement à un sur-apprentissage. Il ne faudrait pas non plus utiliser l'ensemble de test, car il n'est utilisé que pour estimer l'erreur que fait un modèle sur des données qu'il n'a jamais vues, et qui n'ont donc été utilisées ni pour l'optimisation de ses paramètres, ni pour celle de ses hyper-paramètres. Généralement, l'erreur sur l'ensemble d'apprentissage est moins élevée que celle sur l'ensemble de validation car c'est celle qui est directement optimisée par le processus d'apprentissage.

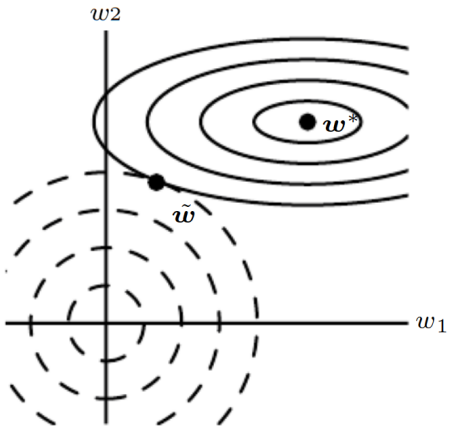

FIGURE 2.5 – Modification du point optimal via la normalisation L^2. Figure extraite de Goodfellow *et al.* [2016]. Les axes w_1 et w_2 représentent deux paramètres à optimiser. Les cercles pleins représentent les courbes de niveaux de la fonction d'erreur non régularisée, atteignant un minimum global en w^*. Les cercles pointillés représentent les courbes de niveau du régulariseur de norme L^2. Ces deux objectifs atteignent leur équilibre au point $\tilde{w}$. La somme des deux erreurs y est minimale. L'erreur non régularisé n'est pas sensible à une variation du paramètre w_1, $\tilde{w}$ a donc pu être amené proche de l'origine dans cette dimension. La différence avec w^* est plus faible pour la dimension w_2 car la fonction d'erreur y est plus sensible (les courbes de niveaux sont plus proches les unes des autres).

Cependant, les jeux de données sont parfois trop petits pour pouvoir être séparés en trois ensembles. Un ensemble d'apprentissage trop petit mène à un sous-apprentissage. Un ensemble de validation trop petit ne permet pas de s'assurer que les hyper-paramètres choisis sont optimaux. Un ensemble de test trop petit ne permet pas de généraliser l'erreur de test à l'estimation de l'erreur de production. Dans ce cas, on peut avoir recours à la validation croisée. Ce processus consiste à ne pas utiliser d'ensemble de test ou de validation, mais d'utiliser tour à tour différentes parties de l'ensemble d'apprentissage en tant qu'ensemble de test. La moyenne des erreurs de test est considérée comme l'erreur de test finale. La validation croisée amène un coût de calcul non négligeable, puisqu'il faut recommencer l'apprentissage chaque fois qu'une différente partie de l'ensemble d'apprentissage est utilisée comme ensemble de test. Nous verrons dans le Chapitre 3 que ce coût en temps de calcul est trop élevé pour que nous puissions appliquer une validation croisée. Ce procédé est illustrée en Figure 2.6.

2.2.3.2 Optimisation

Pour obtenir les hyper-paramètres optimaux pour un modèle donné appliqué à un problème donné, il est possible de se baser sur les conseils d'experts dans le

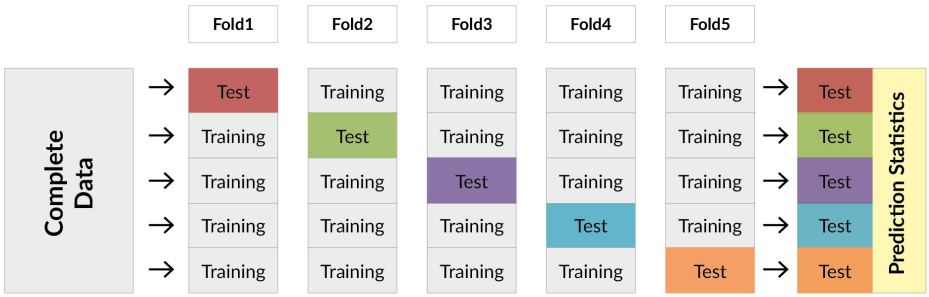

FIGURE 2.6 – Validation croisée à 5 échantillons, dite *5-folds*.

domaine, ou sur la similitude du problème traité à d'autres problèmes pour lesquels les hyper-paramètres optimaux sont connus. Lorsqu'une telle connaissance n'est pas disponible, il est nécessaire d'appliquer un algorithme d'optimisation.

Souvent, les hyper-paramètres ne peuvent pas être optimisés comme les paramètres par de l'apprentissage parce que la fonction donnant la valeur de l'erreur en fonction des hyper-paramètres n'est pas connue ou est trop coûteuse à évaluer. Dans le cas contraire, il sera possible d'utiliser un algorithme basé sur la descente du gradient [Larsen *et al.*, 1996; Chapelle *et al.*, 2002; Foo *et al.*, 2008].

Quand ces méthodes ne sont pas applicables, des algorithmes alternatifs sont utilisés. Le plus naïf de ces algorithmes est la recherche dite *en grille*. Il s'agit d'évaluer le modèle pour toutes les combinaisons possibles d'hyper-paramètres dans un ensemble de valeurs données. Si on reprend l'exemple de la combinaison linéaire des x_i^i évaluée avec une RMSE régularisée, vu en Section 2.2.2, on obtient deux hyper-paramètres à optimiser : c le degré du polynôme et λ le coefficient de régularisation. Pour la recherche en grille, il faut d'abord décider arbitrairement des valeurs possibles pour chacun d'entre eux. Disons $c \in \{1, 2, 3, 4\}$ et $\lambda \in \{0.01, 0.03, 0.1, 0.3, 1, 3\}$. Cela nous donne un total de 24 combinaisons à évaluer. Ce n'est réaliste que si l'apprentissage du modèle est très rapide, ce qui sera le cas ici avec un modèle linéaire et si peu de paramètres à optimiser.

Parcourir l'ensemble de l'espace de recherche peut vite devenir problématique lorsque la durée d'apprentissage est longue ou que le nombre de dimensions de cet espace grandit. La recherche aléatoire consiste à tirer aléatoirement un sous ensemble des combinaisons possibles. Elle ne garantit pas d'arriver à un résultat optimal mais permet de réduire grandement le nombre d'apprentissages nécessaire, ce qui en fait une alternative viable à la recherche en grille. Le problème de cette méthode est qu'elle n'utilise pas le résultat de l'évaluation du tirage précédent pour guider le prochain tirage. Il va donc souvent arriver que la combinaison tirée donne un résultat mauvais qui était prévisible, gâchant alors un temps d'apprentissage précieux.

Pour surmonter cette limite on peut choisr l'optimisation Bayésienne. C'est par exemple celle utilisée par défaut à Google [2]. Cette optimisation vise à découvrir des

2. https://cloud.google.com/blog/products/gcp/

extremums du coût en fonction des hyper paramètres avec le moins de combinaisons testées possible. Pour atteindre cette efficacité, on utilise une fonction d'acquisition. C'est un compromis fait de manière automatique entre l'exploration (découvrir l'erreur à un point de l'espace des hyper-paramètres où la valeur de la fonction de coût est très incertaine) et l'exploitation des résultats connus (découvrir l'erreur à un point de l'espace des hyper-paramètres où on s'attend à ce que le coût soit très bas). Cet algorithme étant utilisé pour des problèmes de maximisation plutôt que de minimisation, on cherchera à maximiser l'opposé du coût plutôt qu'à minimiser le coût. Brochu *et al.* [2010] fournissent une explication détaillée de cette méthode.

D'autres techniques existent, parmi lesquelles la stratégie d'évolution CMA-ES [Igel *et al.*, 2007] ou des algorithmes génétiques [Di Francescomarino *et al.*, 2018], mais n'ont jamais été utilisées dans le cadre de cette thèse.

2.3 Apprentissage profond

L'apprentissage profond regroupe une famille de modèles d'apprentissage automatique, décrits sous forme de réseaux de neurones organisés en couches. Dans nos travaux, l'ensemble des modèles d'apprentissage supervisé pour la segmentation d'images sont des réseaux de neurones artificiels, utilisés pour faire de l'apprentissage profond. Nous verrons en Section 2.4 que l'apprentissage profond peut également être non supervisé.

2.3.1 Principes généraux

Un réseau de neurones est un modèle d'apprentissage automatique constitué d'unités de calculs appelés neurones. Un neurone calcule une somme pondérée de ses entrées et applique une fonction non linéaire, appelée fonction d'activation, au résultat de cette somme. Dans le cas d'un réseau de neurones à propagation avant, la sortie de ce neurone est transmise en entrée d'autres neurones, jusqu'à atteindre la sortie du réseau. Cette propagation itérative crée le concept de couches. Dans le cas d'un réseau de neurones récurrent, la sortie d'un neurone peut être réutilisé en entrée de ce même neurone. Nous n'étudierons que le cas des réseaux de neurones à propagation avant, seul type de réseau de neurones utilisé durant cette thèse. L'apprentissage d'un réseau de neurones consiste à optimiser la pondération de chaque somme. Ces poids sont les paramètres du réseau. Les différentes politiques d'optimisation des paramètres sont mises en application grâce à l'algorithme de rétro-propagation [Rumelhart *et al.*, 1986]. La section 2.3.1.1 présente le fonctionnement d'un neurone, notamment l'intérêt de combiner plusieurs neurones et les fonctions d'activations utilisées. La section 2.3.1.2 présente l'algorithme utilisé pour appliquer les politiques d'optimisation des paramètres.

2.3.1.1 Fonctionnement d'un neurone

L'inspiration biologique du neurone artificiel est relative. Elle s'appuie sur un modèle très simplifié du neurone biologique, illustré en Figure 2.7. Les dendrites

sont les entrées du neurone. D'autres neurones envoient un signal chimique vers les synapses de ces dendrites (leurs entrées), qui le transforme en signal électrique. Les synapses ont une action excitatrice ou inhibitrice. La sortie du neurone artificiel correspond au cône d'émergence du neurone biologique, situé au début de l'axone. Si, ensemble, les dendrites reçoivent un signal assez fort, alors le neurone sera activé. Quand il est activé, il transmet un courant électrique le long de l'axone, libérant des neurotransmetteurs au bouton terminal de l'axone.

De la même manière, un neurone artificiel reçoit des signaux d'autres neurones en entrée. Les synapses sont modélisées par des poids attribués aux valeurs d'entrée. Plus le poids est positif plus l'entrée a une action excitatrice, plus le poids est négatif plus l'entrée a une action inhibitrice. Si la somme pondérée de ces entrées dépasse un certain seuil, le neurone est activé. La fonction décidant de ce seuil est la fonction d'activation. Le résultat de cette fonction est transmis aux neurones suivants.

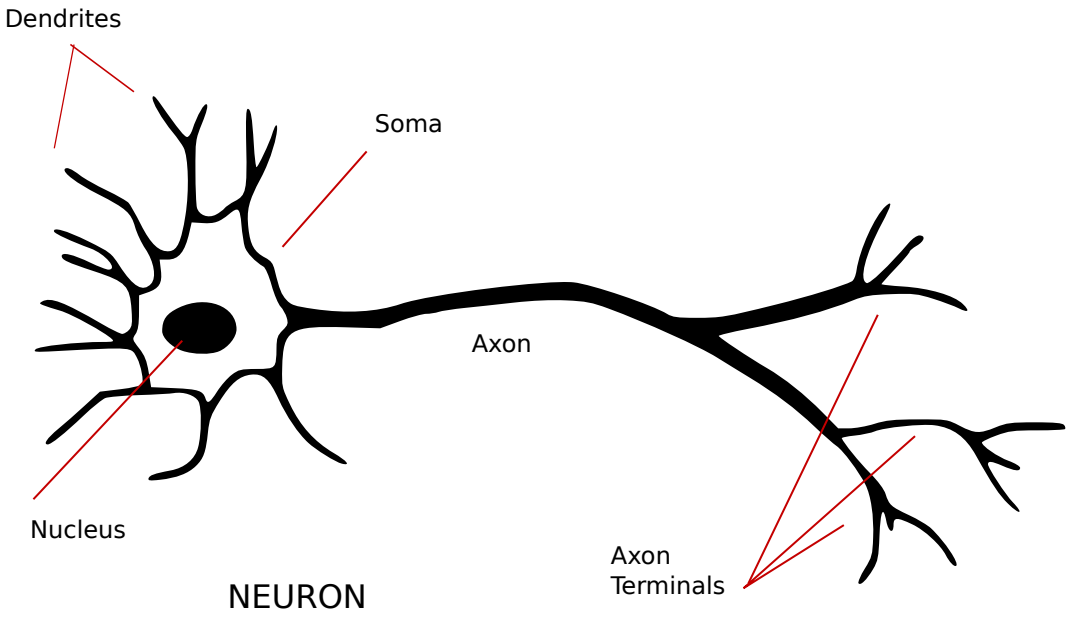

FIGURE 2.7 – Schéma de neurone biologique simplifié, extrait de wikimedia [3]

Le Perceptron [Rosenblatt, 1957] est le premier neurone artificiel qui fut utilisé pour la reconnaissance de formes. Sa première implémentation est matérielle (donc non entraînable), des transistors étant utilisés pour pondérer la somme des entrées. Il permet de procéder à une classification binaire quand les données sont linéairement séparables.

La fonction d'activation du perceptron est la fonction de Heaviside (voir Équation 2.16).

3. https://commons.wikimedia.org

$$\forall x \in \mathbb{R},\ H(x) = \begin{cases} 0 & \text{si} \quad x < 0 \\ 1 & \text{si} \quad x \geq 0. \end{cases} \tag{2.16}$$

La formule du perceptron est donc :

$$f(\boldsymbol{x}) = H(\boldsymbol{x}^T \boldsymbol{w}) \tag{2.17}$$

où $\boldsymbol{w}$ est le vecteur de poids du neurone.

Deux problèmes empêcheront ce modèle d'être largement utilisé. Premièrement, il ne peut être appliqué que sur des données linéairement séparables. En effet, Minsky [1969] montre que la fonction XOR, illustrée en Figure 2.8, ne peut être apprise par un simple perceptron. Il faut utiliser un perceptron multi-couches pour résoudre ce problème, c'est à dire un réseau de neurones et non pas un simple neurone. La modélisation de la fonction XOR est illustrée en Figure 2.9. On notera que chaque couche comporte un neurone de biais. C'est un neurone dont l'activation vaut toujours 1.

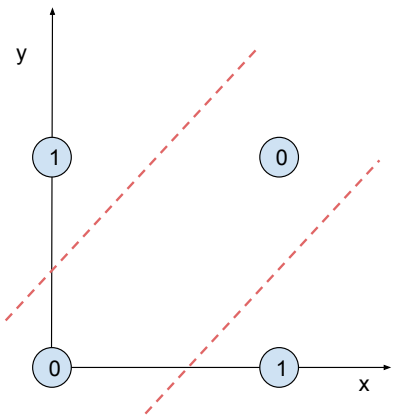

FIGURE 2.8 – Fonction logique XOR. On voit ici qu'il faut utiliser deux droites pour séparer les valeurs 1 des valeurs 0. Elles ne sont pas linéairement séparables.

Le deuxième problème vient de la non continuité de la fonction d'activation illustrée en Figure 2.10. On a vu en section 2.1.3 que l'algorithme de descente du gradient demande à ce que l'on puisse estimer la valeur de la fonction f utilisée lorsqu'on change la valeur de son entrée de manière infinitésimale ϵ : $f(x+\epsilon) \approx f(x) + f'(x+\epsilon)$. Or, ce n'est pas le cas de la fonction de Heaviside H pour laquelle $H(0-\epsilon/2) = 0$ et $H(0+\epsilon/2) = 1$. De ce fait, entraîner un perceptron multi-couches ne peut pas être fait grâce à la descente du gradient, utilisée dans l'algorithme de rétro-propagation présenté dans la section suivante.

Les caractéristiques recherchées dans une fonction d'activation sont les suivantes :

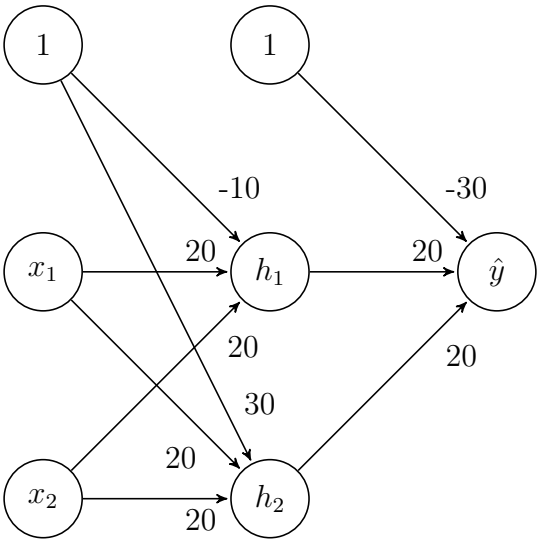

FIGURE 2.9 – Réseau de neurones modélisant un XOR. Tous les neurones sont des perceptrons. Le réseau est de profondeur 3, avec une couche d'entrée comprenant les entrées x_1 et x_2, ainsi qu'un neurone de biais dont la valeur vaut toujours 1. La deuxième couche est appelée une couche cachée, ce n'est ni une couche d'entrée ni une couche de sortie. Elle contient un neurone de biais, et deux neurones intermédiaires h_1 et h_2. La couche de sortie ne contient qu'un neurone $\hat{y}$ donnant le résultat de l'opération $XOR(x_1, x_2)$

- La non linéarité. Elle permet, en combinant plusieurs neurones, de séparer des données qui ne sont pas linéairement séparables. C'est nécessaire par exemple pour modéliser la fonction XOR.
- La dérivabilité. Elle permet l'utilisation d'un apprentissage basé sur la descente du gradient [LeCun *et al.*, 1989].
- La monotonie. Elle permet d'éviter les minimums locaux [Csáji, 2001].
- L'identité en 0. Elle permet une initialisation aléatoire des poids. [Sussillo, 2014]

Les trois principales fonctions d'activation de la litérature sont les suivantes : sigmoïde, tanh, et ReLU, tracées en Figure 2.11. Sigmoïde est une classe de fonctions ayant une forme de S, avec une asymptote horizontale aux infinis. Lorsqu'on parle de la fonction sigmoïde, on fait par défaut référence à la fonction logistique :

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{2.18}$$

Elle permet de régler le problème de non dérivabilité de la fonction de Heaviside. Elle connaît une asymptote horizontale d'équation $y = 0$ pour $x \to -\infty$ et une asymptote horizontale d'équation $y = 1$ pour $x \to \infty$.

Cette fonction présente deux problèmes. En dehors de l'intervalle $[-3; 3]$, la dérivée de la sigmoïde est quasi nulle, ce qui peut empêcher l'apprentissage (voir Section 2.3.1.2). Aussi, la fonction sigmoïde ne produit pas de valeur négative. Or, LeCun *et al.* [2012] explique que pour un apprentissage plus rapide, la sortie moyenne d'un

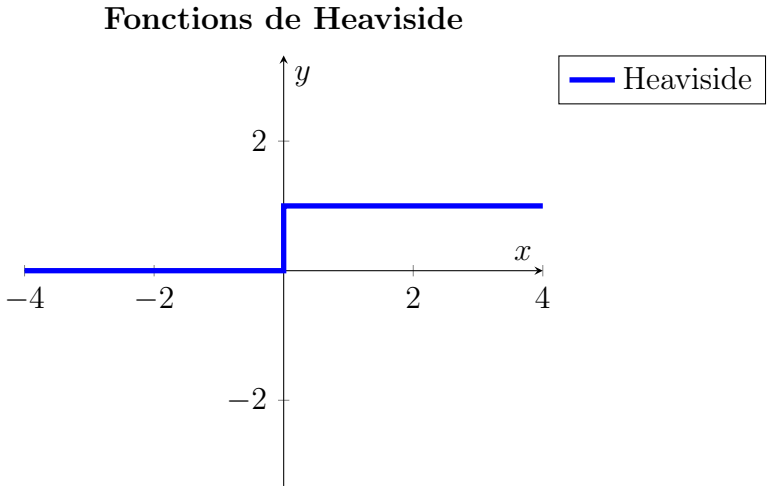

FIGURE 2.10 – tracé de la fonction de Heaviside

neurone sur tous les exemples de l'ensemble d'apprentissage devrait être nulle. En effet, lors de l'apprentissage les poids du neurone sont mis à jour de manière proportionnelle à l'erreur et à l'entrée (voir Section 2.3.1.2). Si l'entrée n'est constituée que de valeurs positives, la mise à jour des poids sera systématiquement du signe de l'erreur. On verra donc tous les poids croître ou tous les poids décroître quand il conviendrait d'en faire croître certains et décroître d'autres. Pour éviter ce cas de figure, une moyenne nulle des entrées assure d'en avoir certaines positives et certaines négatives.

La fonction d'activation *tanh* résout ce problème. C'est une sigmoïde ayant ses images dans $[-1; 1]$:

$$tanh(x) = 2 * sigmoid(2 * x) - 1 \tag{2.19}$$

Cette fonction faisant partie de la famille des sigmoïdes, le problème de gradient nul pour de grandes valeurs subsiste. Contraindre les valeurs traversant le réseau à être petites, grâce à la normalisation par exemple, ne permet pas de résoudre ce problème. En réalité, si les valeurs sont trop petites alors la fonction *tanh* est proche de la fonction identité, elle présente une asymptote d'équation $y = x$ pour $x \to 0$. Or, la fonction identité n'est pas une fonction d'activation intéressante parce qu'elle est linéaire.

La fonction ReLU présente elle une non linéarité autour de 0 :

$$ReLU(x) = \begin{cases} 0 & \text{pour} \quad x < 0 \\ x & \text{pour} \quad x \geq 0 \end{cases} \tag{2.20}$$

C'est la fonction utilisée par défaut aujourd'hui. Elle présente plusieurs avantages. Sa non linéarité pour les petites valeurs permet la normalisation de ses entrées. Sa dérivée est très simple ce qui permet des calculs très rapide et donc un temps

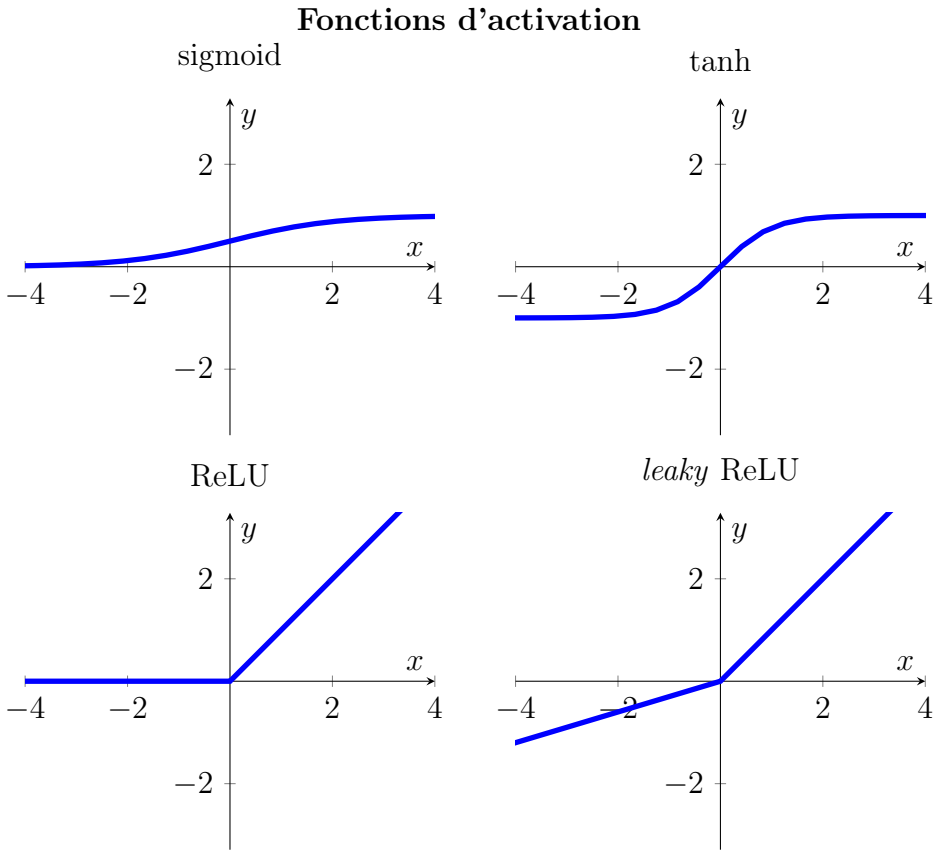

FIGURE 2.11 – Les courbes des quatre principales fonctions d'activation. la pente du *leaky* ReLU a été exagérée pour plus de visibilité.

d'apprentissage réduit (la dérivée en 0 est définie comme nulle). Elle ne présente pas l'inconvénient de n'avoir que des valeurs strictement positives, donc faire varier des poids dans un sens n'oblige pas à faire varier tous les poids dans ce sens. Elle est éparse, ce qui signifie qu'elle n'est pas activée en permanence. En effet pour toutes les valeurs négatives cette fonction n'active pas le neurone. Cela permet d'avoir des rôles bien définis pour chaque neurone, qui peut ne s'activer que si besoin est. Cependant, puisqu'une dérivée nulle peut poser problème lors de l'apprentissage, une version améliorée du ReLU existe, le *leaky* ReLU :

$$ReLU_{leaky}(x) = \begin{cases} 0.01x & \text{pour} \quad x < 0 \\ x & \text{pour} \quad x \geq 0 \end{cases} \tag{2.21}$$

Les courbes de ces fonctions d'activation sont tracées en Figure 2.11 et celles de leurs dérivées en Figure 2.12.

La fonction d'activation de la couche de sortie dépend du type de problème que l'on veut résoudre. Lorsqu'on souhaite prédire si oui ou non l'objet étudié appartient à une classe donnée, on choisira généralement une fonction sigmoïde, nous donnant

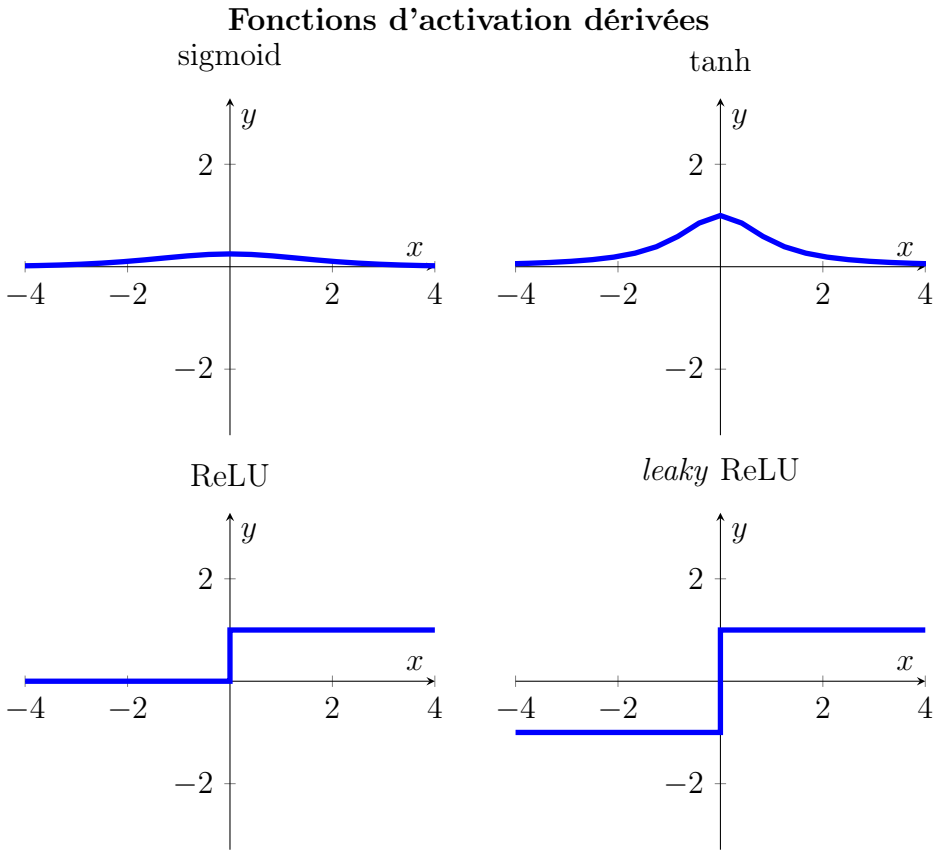

FIGURE 2.12 – Les courbes des dérivées des trois principales fonctions d'activation. La courbe de la dérivée du *leaky* ReLU' est normalement presque confondue avec celle du ReLU', les valeurs pour $x < 0$ ont été ramenées à -1 pour plus de visibilité.

une probabilité. Si on veut déterminer la classe de l'objet étudié parmi un panel de K classes possibles, on choisira généralement un *softmax*, défini par l'équation 2.22. Cette fonction prend en entrée $\mathbf{x}$ un vecteur de réels de dimension K et produit un vecteur de réels strictement positifs, de dimension K, dont la somme des valeurs vaut 1. On obtient ainsi la probabilité pour l'objet d'appartenir à chacune des classes. Sa formule est la suivante :

$$softmax(\mathbf{x})_j = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}} \forall j \in \{1, \ldots, K\} \tag{2.22}$$

2.3.1.2 Rétro propagation

La rétro-propagation [Rumelhart *et al.*, 1986; LeCun *et al.*, 1989] est l'algorithme permettant d'appliquer la politique d'apprentissage aux poids du réseau de neurones. Le but est de pouvoir appliquer une forme de descente du gradient au réseau. Il faut pour cela pouvoir déterminer la dérivée partielle de la fonction de coût en fonction

de chacun des paramètres du réseau.

Les notations utilisées pour décrire l'algorithme de rétro-propagation sont les suivantes. Tout d'abord, il est nécessaire de nommer les paramètres du réseau de manière non ambiguë. Le poids sur l'arête reliant le k-ième des K neurones de la $(l-1)$-ième des L couches, au j-ième des J neurones de la couche l est nommé w^l_{jk}. Le biais du k-ième neurone de la couche l est noté b^l_k et son activation a^l_k. La Figure 2.13 montre un exemple d'appellations dans un réseau de neurones. On rappelle qu'un neurone de biais est présent à chaque couche, c'est un neurone dont l'activation vaut toujours 1.

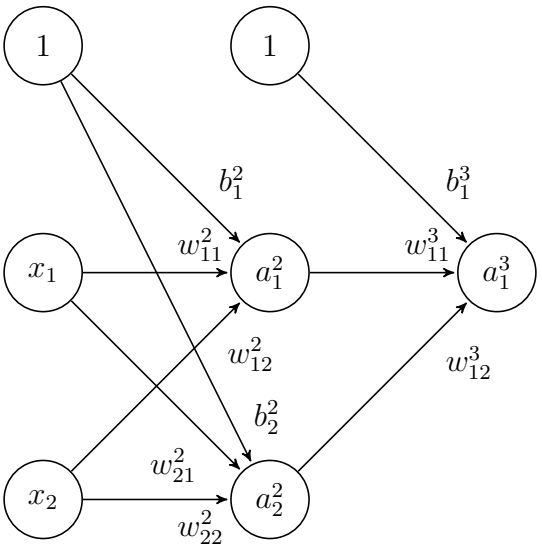

FIGURE 2.13 – Réseau de neurones,$L = 3$. Les arêtes sont nommées conformément à la notation décrite en Section 2.3.1.2.

On sait donc que pour un coût C donné, on cherche à déterminer $\frac{\partial C}{\partial w^l_{jk}}$ la dérivée partielle du coût pour chaque poids et $\frac{\partial C}{\partial b^l_j}$ la dérivée partielle du coût pour chaque biais.

Étant donné une fonction d'activation σ, on obtient donc que l'activation du j-ième neurone de la l-ième couche est défini par l'équation suivante, où k désigne chaque neurone de la couche $l-1$, sachant que l'activation de la première couche correspond aux données d'entrée :

$$a^l_j = \sigma\left(\sum_k w^l_{jk} a^{l-1}_k + b^l_j\right) \tag{2.23}$$

On peut, pour simplifier la notation, décrire ces opérations sous forme matricielle. La matrice $\boldsymbol{W}^l$ est la matrice contenant les poids reliant la couche $l-1$ à la couche l, large de K colonnes (le nombre de neurones de la couche $l-1$) et haute de J lignes (le nombre de neurones de la couche l). On trouve à la j-ième ligne et k-ième

colonne de $\boldsymbol{W}^l$ le poids w^l_{jk}. De la même manière $\boldsymbol{b}^l$ et $\boldsymbol{a}^l$ désignent respectivement les vecteurs contenant les biais et les activations des neurones de la l-ième couche. Ces deux vecteurs sont de taille J.

En considérant que la fonction d'activation vectorisée est simplement la fonction d'activation appliquée à chaque élément d'un vecteur un à un ($\sigma(\boldsymbol{x})_j = \sigma(x_j)$), on obtient alors la version vectorisée de l'équation précédente :

$$\boldsymbol{a}^l = \sigma\left(\boldsymbol{W}^l * \boldsymbol{a}^{l-1} + \boldsymbol{b}^l\right) \tag{2.24}$$

Pour obtenir l'activation a^l_j on applique la fonction d'activation prédéfinie au résultat de la somme pondérée des entrées. Ce résultat est noté z^l_j. L'erreur associée à cette activation est notée δ^l_j et a pour équation

$$\delta^l_j = \frac{\partial C}{\partial z^l_j} \tag{2.25}$$

L'algorithme de rétro-propagation suppose que la fonction de coût puisse être exprimée comme la moyenne des coûts sur l'ensemble des exemples, et que son entrée corresponde au vecteur d'activation de la dernière couche du réseau. Cette deuxième condition nous permet de définir $\nabla_{\boldsymbol{a}}\boldsymbol{C}$ comme étant le vecteur contenant les dérivées partielles de la fonction de coût par rapport à l'activation de chaque neurone de la couche de sortie L : $\frac{\partial C}{\partial a^L_j}$. Avec cette définition on pose l'équation suivante :

$$\boldsymbol{\delta}^L = \nabla_{\boldsymbol{a}}\boldsymbol{C} \odot \sigma'\left(\boldsymbol{z}^L\right) \tag{2.26}$$

L'opérateur $\odot$ désigne le produit terme à terme de deux matrices ou vecteurs. Tous les éléments à droite de cette équation sont connus et permettent de déterminer l'erreur de la dernière couche du réseau. Il faut maintenant propager cette erreur à la couche précédente :

$$\boldsymbol{\delta}^l = \left(\left(\boldsymbol{W}^{l+1}\right)^T \boldsymbol{\delta}^{l+1}\right) \odot \sigma'\left(\boldsymbol{z}^l\right) \tag{2.27}$$

Où σ' est la dérivée de la fonction d'activation. Cette équation permet de calculer l'erreur de la couche précédent n'importe quelle couche dont on connaît l'erreur, les poids, la fonction d'activation commune à tous ses neurones, et les valeurs reçues en entrées.

On peut donc maintenant calculer les valeurs initialement recherchées $\frac{\partial C}{\partial w^l_{jk}}$ et $\frac{\partial C}{\partial b^l_j}$:

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j \tag{2.28}$$

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j \tag{2.29}$$

Connaissant la dérivée partielles de l'erreur par rapport à chaque paramètre du réseau, on peut appliquer pour chacun d'entre eux un algorithme basé sur la descente du gradient pour optimiser les poids du réseau.

Le problème de fuite du gradient est lié à l'algorithme de rétro propagation et aux fonctions d'activation choisies. Les fonctions de type sigmoïde ne permettent pas de restreindre par la normalisation les valeurs d'entrées à être proche de 0. En effet, au voisinage de 0 ces fonctions sont quasi linéaires. Cependant, si ces fonctions ont un large domaine de définition ($]-\infty;+\infty[$), elles ont une image dans un domaine restreint. Pour la fonction *tanh* par exemple, l'ensemble des valeurs dans $]-\infty;-3]$ ont leur image comprise dans $]-1;-0.995]$. Ce qui veut dire que la probabilité qu'une large variation de la valeur d'entrée n'entraîne qu'une très faible variation de la valeur de sortie est très grande.

Pour que le modèle apprenne, il faudra que l'algorithme de mise à jour des paramètres soit capable de les faire varier suffisamment pour obtenir une réponse du réseau différente. Or, si on observe l'Équation 2.27 on voit que l'erreur utilisée pour déterminer la dérivée partielle par rapport à un paramètre est proportionnelle à la valeur de la dérivée de l'activation en un point. Or on a vu que cette valeur est quasi nulle pour une grande majorité des points des fonctions sigmoïdes.

Ces fonctions d'activation présentent donc de grands risques d'annuler le gradient et d'empêcher le réseau de mettre à jour ses paramètres, et donc d'apprendre. Ce problème est en partie évité par l'utilisation d'autres fonctions d'activation telles que la fonction ReLU, ou par des architectures adaptées telle que l'architecture ResNet, détaillée en Section 3.1.2.2.

Des solutions alternatives à la rétro-propagation existent, telles que les méthodes PowerSign et AddSign, proposées par Bello *et al.* [2017]. Elles utilisent un réseau de neurones qui apprend à optimiser un autre réseau de neurones. Les deux méthodes découvertes par ce premier réseau leur permettent d'atteindre un état final plus optimal que les algorithmes classiques sur les problèmes de classification habituellement utilisés (ImageNet et CIFAR-10). Cependant, la rétro-propagation reste la méthode conventionnelle d'optimisation pour le moment.

2.3.2 Réseaux convolutifs

Lorsqu'on traite une image, les propriétés données en entrée du réseau de neurones sont les valeurs décrivant chaque pixel (dans le cas d'une image RGB ce seront 3 valeurs, la quantité de rouge, de vert et de bleu). Quand l'image est petite, et centrée sur l'objet d'intérêt comme dans le jeu de données MNIST [LeCun *et al.*, 1998], on peut se permettre d'avoir des réseaux entièrement connectés, où tous les neurones d'une couche sont connectés à tous les neurones de la couche suivante, et la première couche contient un neurone par pixel et par couche de l'image. Très rapidement, les images deviennent trop grosses pour ce type de réseau, le nombre de paramètres explose et le réseau ne tient plus en mémoire. Pour palier à ce problème, on utilise un réseau de taille réduite, traitant chaque pixel et son voisinage un par un à l'aide de convolutions. Les convolutions servent à fabriquer des cartes de propriétés, sorte d'images multi-couches, dont chaque couche est une propriété obtenue grâce à un filtre de convolution, et chaque localité (l'équivalent des pixels pour une image) la valeur de la propriété à cet endroit. Alliées à des procédés de réduction de dimensionnalité, les cartes de propriétés sont de tailles réduites et peuvent être

ensuite traitées par un réseau entièrement connecté. On appelle ce type de modèle un réseau de neurones convolutifs, ou CNN, de l'anglais *Convolutional Neural Network*.

2.3.2.1 Convolutions

Le type de couche primordial utilisé dans un CNN est la couche de convolution. Dans un réseau de neurones tel qu'on les a présentés précédemment, traverser une couche correspond à multiplier le vecteur contenant les valeurs entrantes avec la matrice des poids, puis appliquer une transformation non linéaire au vecteur résultant. La disposition des données entrante importe peu tant qu'elle est constante. Lorsqu'on traite une image (mais aussi un son, du texte...) la position du pixel dans l'image et par rapport à ses voisins a une importance. Une couche de convolution va pouvoir tirer partie de cette information.

La fenêtre (ou noyau) de convolution, est une matrice généralement carrée et de petite taille. Les tailles les plus communes sont des fenêtre de $3 * 3$, cette taille est fixée à la conception du réseau. Elle contient les paramètres de la couche convolutive. Cette matrice a une troisième dimension, correspondant à la profondeur de la carte de propriétés traitée, fixée par le nombre de convolutions de la couche précédente. Cette fenêtre glisse sur la carte de propriété : à chaque emplacement, on calcule le produit terme à terme entre la fenêtre de convolution et la partie de la carte de propriétés sur laquelle elle se trouve. Une fonction non linéaire est appliquée sur la somme des éléments de la matrice résultante et la valeur ainsi obtenue correspondra à la valeur d'une localité au sein d'une couche de la carte de propriété résultante. La Figure 2.14 illustre ce phénomène. Les paramètres contenus dans la matrice convoluée sont les poids du réseau qui seront optimisés lors de l'apprentissage. Ils ne changent pas pendant le processus de convolution, chaque emplacement est traité avec les mêmes poids. Cette invariance des poids permet un apprentissage rapide du réseau car il permet de mettre à profit la puissance de calcul des GPUs modernes.

La carte de propriétés résultante a une profondeur arbitraire fixée lors de la conception du réseau. On la choisi en fixant le nombre de fenêtres de convolution qui seront appliquées à la carte de propriétés d'entrée. Le résultat de chaque convolution correspond à une couche dans la carte de propriétés de sortie. Le pas de convolution est aussi choisi lors de la conception du réseau. Il régit le glissement de la fenêtre de convolution. Lorsqu'un emplacement a été traité, la fenêtre se déplace d'un nombre de pixels correspondant au pas.

La taille de la fenêtre de convolution est impaire. On s'intéresse au pixel central et on s'appuie pour cela d'un nombre fixe de pixels dans chaque direction. Lorsque ce pixel central se trouve proche d'un bord de la carte de propriété d'entrée, il peut arriver que la fenêtre de convolution déborde de la carte de propriété. On doit choisir entre ne pas traiter ces pixels, ou ajouter une marge à la carte de propriétés d'entrée de manière à ce que le cas ne se produise pas. Cette marge peut ne contenir que des pixels nuls, ou être construite à partir de la carte de propriétés d'entrée, typiquement grâce à des symétries.

Les paramètres de taille à fixer sont donc :

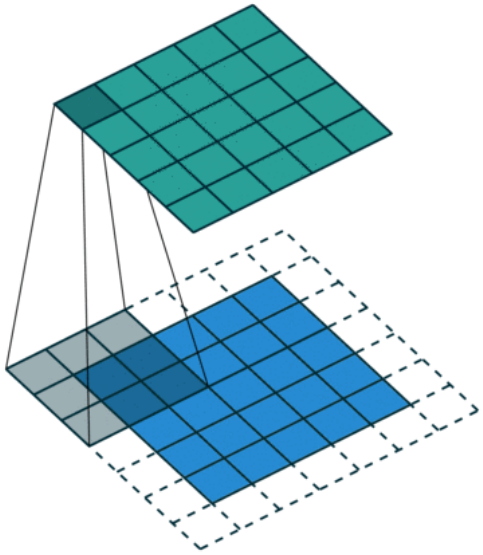

FIGURE 2.14 – Convolution. Figure extraite de Dumoulin et Visin [2016]. On a ici un filtre de convolution de taille 3 par 3 qui vient griser les pixels de la carte de propriété d'entrée (bleue) auquel il est appliqué. la fenêtre de convolution glisse de pixel en pixel pour les traiter un à un. À la fin du processus de convolution, on obtient une couche de la carte de propriété de sortie, en vert. On observe en blanc une marge de pixels nuls, ajoutés de manière à ce que la convolution conserve la taille de carte de propriétés d'entrée.

- La taille de la fenêtre de convolution $((2r_h + 1) * (2r_l + 1))$
- La profondeur de la carte de propriétés sortante, en anglais *depth* (d), correspond au nombre de couches de convolution.
- le pas de convolution, en anglais *stride* (s)
- la taille de la marge appliquée à la carte de propriétés d'entrée, en anglais *padding* (p)

Ces paramètres étant fixés en dehors de tout processus d'apprentissage, ce sont des hyper-paramètres, tels qu'abordés en Section 2.2.3. Ils suffisent à calculer la taille de la carte de propriété de sortie $(h_{out}, l_{out}, d_{out})$ en fonction de la taille de la carte de propriété d'entrée (h_{in}, l_{in}, d_{in}) :

$$(h_{out}, l_{out}, d_{out}) = ((h_{in} - 2r_h + 2p)/s, (l_{in} - 2r_l + 2p)/s, d) \tag{2.30}$$

Plus formellement, en gardant les notations définies précédemment pour les hyper-paramètres, la carte de propriétés F résultant de la convolution de la fenêtre K sur une image I s'exprime suivant l'équation suivante :

$$F_{i,j} = K \star I_{i,j} = \sum_{m \in [-r_h; +r_h]} \sum_{\substack{n \in [-r_l; +r_l] \\ \forall i \in [0; h_{out}], \forall j \in [0; l_{out}]}} I_{s*i+m, s*j+n} K_{m,n} \tag{2.31}$$

On voit qu'un pas de convolution supérieur à 1 résulte en une réduction de la résolution de la carte de propriétés. Ce paramètre est en général fixé à 1, car on préférera utiliser une couche de *pooling* pour effectuer ce sous échantillonnage. Cette couche est l'objet de la Section 2.3.2.2 suivante.

La rétro-propagation est affectée de la manière suivante par la convolution ; en reprenant les notations de l'équation 2.31, nous souhaitons déterminer $\frac{\partial C}{\partial K^l_{m',n'}}$ où m' et n' sont les indices de la fenêtre de convolutions (à différencier de m et n qui en sont les itérateurs dans l'équation 2.31).

Comme pour la rétro-propagation habituelle, par analogie à l'équation 2.25, on note l'erreur δ^l associée à l'activation de la carte de propriétés de la couche l, $F^l_{i,j}$, indexée par i et j :

$$\frac{\delta C}{\delta F^l_{i,j}} = \delta^l_{i,j} \tag{2.32}$$

On écrit alors :

$$\frac{\partial C}{\partial K^l_{m',n'}} = \sum_{i=0}^{h_{out}} \sum_{j=0}^{l_{out}} \frac{\delta C}{\delta F^l_{i,j}} \frac{\partial F^l_{i,j}}{\partial K^l_{m',n'}} = \sum_{i=0}^{h_{out}} \sum_{j=0}^{l_{out}} \delta^l_{i,j} \frac{\partial F^l_{i,j}}{\partial K^l_{m',n'}} \tag{2.33}$$

Nous itérons ici sur l'espace de sortie (la carte de propriétés $\boldsymbol{F}$ de hauteur h_{out} et de largeur l_{out}), déterminons l'erreur associée à l'activation et déterminons ensuite le facteur de contribution du poids du noyau par rapport à cette sortie.

D'après l'équation 2.31 on sait que :

$$\frac{\partial F^l_{i,j}}{\partial K^l_{m',n'}} = \frac{\partial \sum_{m\in[-r_h;+r_h]} \sum_{n\in[-r_l;+r_l]} F^{l-1}_{s*i+m,s*j+n} K_{m,n}}{\partial K^l_{m',n'}} \tag{2.34}$$

La dérivée étant non nulle seulement pour $m = m'$ et $n = n'$, on obtient finalement :

$$\frac{\partial F^l_{i,j}}{\partial K^l_{m',n'}} = F^{l-1}_{i+m',j+n'} \tag{2.35}$$

On trouve finalement l'erreur associée à chaque poids de la fenêtre de convolution :

$$\frac{\partial C}{\partial K^l_{m',n'}} = \sum_{i=0}^{h_{out}} \sum_{j=0}^{l_{out}} \delta^l_{i,j} F^{l-1}_{i+m',j+n'} \tag{2.36}$$

2.3.2.2 Pooling

Le principe du *pooling* est de combiner plusieurs localités d'une carte de propriétés en entrée pour n'en obtenir qu'une en sortie, et ainsi obtenir une carte de propriétés invariante à de petites translations. L'opération de *pooling* la plus commune et le max-*pooling*, qui remplace une zone rectangulaire par la localité de plus haute valeur

qu'elle contient. D'autres opérations peuvent être appliquée, telles que la moyenne (pondérée ou non en fonction la distance à la localité central), la norme L^2 des localités concernées, etc. Cette invariance se gagne au prix de la connaissance de la localisation des objets dans l'image. Cette opération est donc particulièrement pertinente quand connaître ce qui est représenté dans l'image est plus important que de connaître où se situe cette représentation au sein de l'image.

Un autre avantage majeur de cette opération est de réduire la taille des données traitées en mémoire, qui est un facteur limitant dans l'apprentissage d'un CNN. En règle général, les fenêtres de *pooling* sont convoluées sur la carte de propriété d'entrée avec un pas égal à la taille de la fenêtre de convolution. Il n'y a donc ni chevauchement ni espace entre deux emplacements de convolution. La taille de la carte de propriétés résultante de cette opération est donc, en utilisant les mêmes notations que précédemment :

$$(h_{out}, l_{out}, d_{out}) = (h_{in}/s, l_{in}/s, d) \tag{2.37}$$

La plupart du temps, on utilise un max-*pooling* de taille $2 * 2$ avec un pas de 2.

La rétro-propagation est affectée différemment en fonction du *pooling*. Pour un max-*pooling*, le gradient est entièrement attribué à la localité maximale, on attribue une erreur nulle aux autres. Pour un *pooling* qui consisterait à moyenner l'ensemble des pixels d'entrée, on distribue le gradient équitablement entre les entrées, etc.

2.3.2.3 Champs réceptif

Le *pooling* est également une des méthode permettant d'augmenter le champ réceptif d'un neurone. Cette notion désigne la région de la carte de propriété d'entrée ayant permis de déterminer la valeur d'une localité donnée de la carte de propriété de sortie. Le champ réceptif désigne par défaut une région de l'image passée en entrée du réseau et la localité affectée est par défaut une localité en sortie du réseau. On peut toutefois parler du champ réceptif d'une couche l donnée par rapport à une localité d'une couche $l+k$, même si ces couches ne sont pas la première et la dernière. Augmenter le champ réceptif permet de donner plus de contexte pour classifier une localité ainsi que de détecter des objets de plus grande dimension au sein d'une image.

D'autre méthodes que le *pooling* existent pour augmenter le champ réceptif. Chaque couche de convolution augmente linéairement le champ réceptif (chaque couche de convolution augmente la taille du champ réceptif d'une largeur de noyau), alors que le *pooling* l'augmente de manière exponentielle (chaque couche de *pooling* multiplie la taille du champ réceptif par la largeur du noyau). Le champ réceptif peut aussi être augmenté grâce à un pas de convolution supérieur à 1, ou encore, nous le verrons en Section 3.1.2.5, par des agrégations multi-échelles telles que le *pooling* spatial pyramidal et les convolutions dilatées.

Luo *et al.* [2016] montrent que le champ réceptif effectif est différent du champ réceptif théorique. En effet, toutes les localités du champ réceptif ne contribuent pas de manière égale, comme illustré en Figure 2.15.

La fonction d'activation modifie également ce champ effectif, pouvant annuler certaines localités. La fonction ReLU par exemple va clairsemer la distribution gaus-

sienne. L'apprentissage va également modifier le poids indiquant l'importance de chaque localité en fonction du problème traité. Ces modifications sont illustrées au sein des Figures 2.16, 2.17 et 2.18. Dans ces images, plus un pixel est blanc plus il a été utilisé pour construire le pixel final étudié. On voit que la fonction ReLU annule certains pixels, le champ réceptif est clairsemé de pixels noirs. Ce n'est pas le cas des fonctions tanh et sigmoide. L'entraînement permet d'étaler le champ réceptif, le réseau de neurone va chercher plus de contexte, plus loin. Enfin, on voit à quel point le pooling permet d'augmenter la taille du champ réceptif par rapport à une convolution. La dilatation (voir Section 3.1.2.4) permet également un augmentation, de forme rectangulaire.

2.4 Apprentissage non supervisé

À la différence de l'apprentissage supervisé, les données utilisées pour l'apprentissage non supervisé ne sont pas étiquetées. Cela signifie que pour une entrée donnée, on ne sait pas dire quel est le résultat correct, celui qui est juste. L'objectif est induit

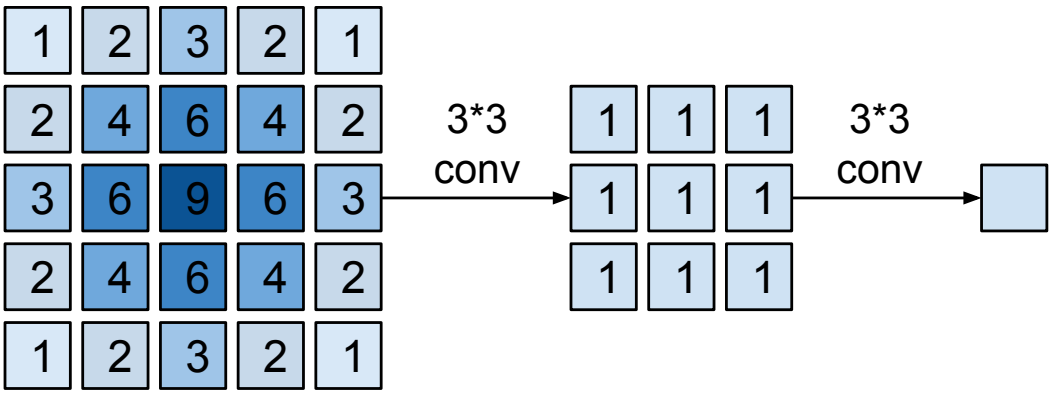

FIGURE 2.15 – Champ réceptif d'une localité de la couche l après deux convolutions de fenêtres $(3 * 3)$ aux couches $l - 1$ et $l - 2$. Le nombre au sein de chaque localité indique le nombre de fois où elle a été utilisée au sein d'une opération de convolution afin de déterminer la valeur de la la localité finale.

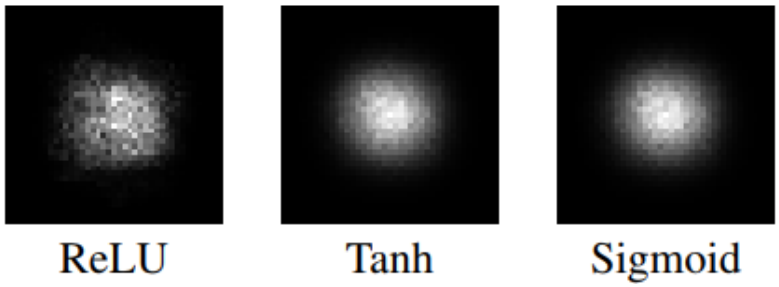

FIGURE 2.16 – Figure extraite de Luo *et al.* [2016]. Effet des fonctions d'activations sur le champ réceptif effectif.

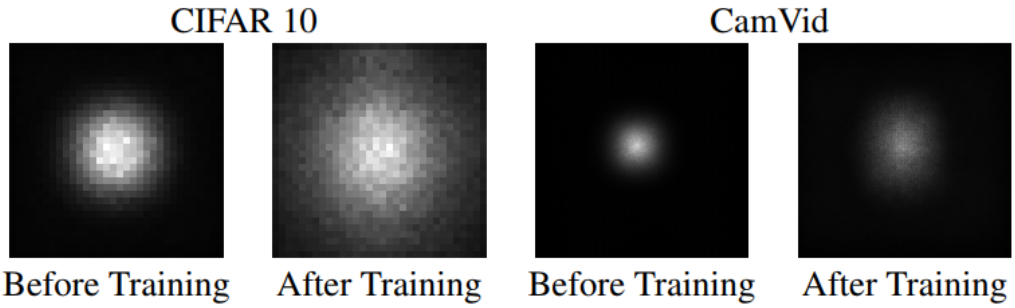

FIGURE 2.17 – Figure extraite de Luo *et al.* [2016]. Effet de l'apprentissage sur le champ réceptif effectif, sur les jeux de données CIFAR 10 et CamVid.

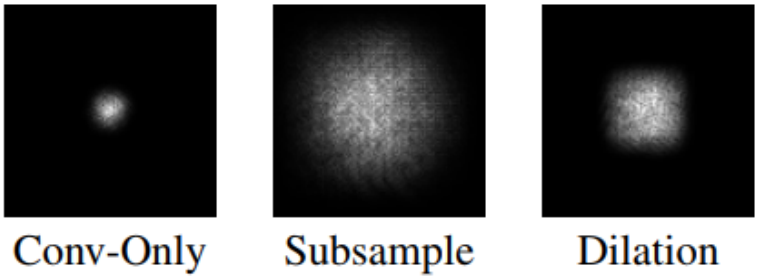

FIGURE 2.18 – Figure extraite de Luo *et al.* [2016]. Effet de différentes convolutions sur le champ réceptif effectif.

par la formulation du problème. On cherche en général à mettre en évidence une structure dans les données. On va chercher par exemple à déterminer la distribution de probabilités ayant permis de générer le jeu de données ou regrouper des données similaires. On s'intéresse ici à trois catégories de modèles. Les modèles de partitionnement, de détection d'anomalies, et quelques réseaux de neurones. Le but de ces modèles est d'apprendre le schéma de distribution interne des données, dans le but de le visualiser, en retirer des informations, ou le reproduire.

2.4.1 Partitionnement

Le partitionnement de données consiste à créer des groupes et distribuer les échantillons de données au sein de ces groupes, de manière à ce que les groupes d'objets partagent des propriétés communes. On définit pour cela une mesure de distance entre deux échantillons. Un bon partitionnement des données minimise les distances au sein d'un groupe tout en maximisant les distances entre groupes.

2.4.1.1 Applications

Les algorithmes de partitionnement peuvent être utilisés pour faire de la détection d'anomalies (voir Section 2.4.2), par exemple en séparant les patients sains des patients atteints de cancer [Girolami et He, 2003; Wang et Garibaldi, 2005], ou en

détectant des intrusions au sein d'un réseau [Portnoy, 2000; Münz *et al.*, 2007]. Ils sont également utilisés pour catégoriser des ensembles. Par exemple, Oyelade *et al.* [2010] séparent les élèves en niveaux de performances, et utilisent le partitionnement K-Means [MacQueen, 1967] plutôt que de définir les niveaux arbitrairement. Hamaguchi et Hikosaka [2018] catégorisent les bâtiments en fonction de leur taille (petits, moyens, grands) afin d'entraîner un modèle de détection pour chacune de ces classes dans le but d'extraire les bâtiments d'images satellites. On utilise aussi le partitionnement pour trouver des éléments similaires. Des algorithmes de recommandation sont basés sur du partitionnement, pour le e-commerce [Sarwar *et al.*, 2002] ou la recommandation de films [Gomez-Uribe et Hunt, 2016].

2.4.1.2 Approches

Les algorithmes de partitionnement sont souvent hiérarchiques, basés sur la distribution ou la densité, ou basés sur des centroïdes. Les partitionnement hiérarchiques font apparaître des partitions au sein d'autres partitions, soit en découpant des partitions en sous partitions, comme le font Dhillon *et al.* [2003] pour de la classification de texte, ou au contraire en agrégeant entre elles des partitions existantes, comme le font Perrot *et al.* [2015] pour visualiser des cartes de densités d'un grand nombre de points. Si on connaît le type de distribution des données, on utilisera plutôt un partitionnement basé distribution, tel que le modèle de mélange gaussiens [Reynolds, 2015], qui gère très bien les partitions se chevauchant, mais très mal les partitions basées sur la densité, qui seront bien mieux traités par un modèle tel que DBSCAN [Ester *et al.*, 1996]. Une comparaison est effectuée en Figure 2.19. Enfin, les partitionnements basés centroïdes, dont le plus utilisé est le K-Means [MacQueen, 1967], permettent de mesurer une similarité entre les éléments. Parce qu'ils sont basés sur une distance (euclidienne dans le cas du K-Means), il peut être difficile de les utiliser si les échantillons contiennent des propriétés catégorielles. Ienco *et al.* [2012] proposent une méthode pour pouvoir calculer une distance sur ces exemples en se basant des mesures de similarité contextuelle.

2.4.2 Détection d'anomalies

Une anomalie est un motif ne se conformant pas aux attentes. C'est un cas particulier, une valeur aberrante. Des méthodes de statistiques peuvent parfois être employées pour détecter des anomalies. Par exemple un point s'éloignant d'un nombre d'écarts types spécifique de la moyenne des points peut être une anomalie. Cependant, ces méthodes peuvent être mises à mal par du bruit dans les données venant cacher les anomalies et par des définition imprécises ou changeantes d'une anomalie en fonction du contexte ou du temps.

Chandola *et al.* [2009] définissent trois catégories d'anomalies. Les anomalies ponctuelles sont les plus communes. Si on peut classifier un point du jeu de données comme anormal par rapport aux autres alors l'anomalie est ponctuelle.

Une anomalie contextuelle est plus subtile, en fonction du contexte un même point peut être considéré comme normal ou anormal. Pour cela, il faut séparer les

4. https://commons.wikimedia.org/

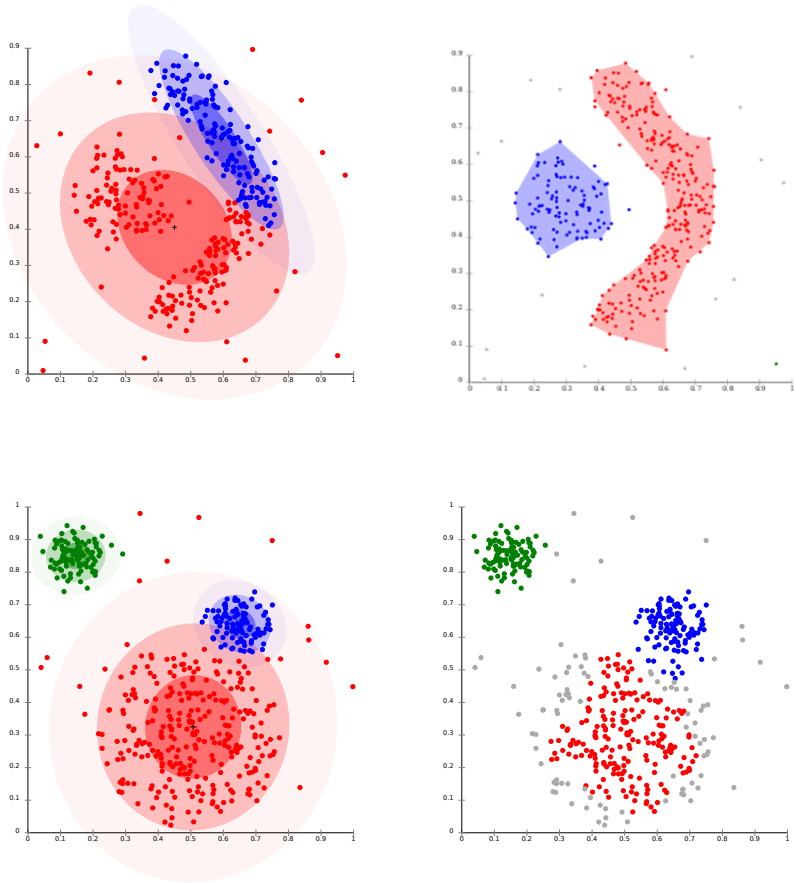

FIGURE 2.19 – Comparaison du partitionnement par modèle de mélange gaussien et par DBSCAN. Les quatre figures proviennent de wikimedia [4]. La première colonne est le résultat de partitionnement par un modèle de mélange gaussien. Quand la distribution des échantillons au sein de la partition est effectivement gaussienne, ce modèle donne des résultats qualitativement intéressants, mais échoue à partitionner des données ne suivant pas cette distribution. Au contraire, sur la deuxième colonne on voit que DBSCAN donne une bonne partition basée sur la densité dans le premier cas, mais tend à faire des partitions de taille équivalente et ne parvient pas à délimiter des partitions quand la densité ne change pas brutalement.

propriétés d'un point en deux types. Les propriétés contextuelles d'un point peuvent typiquement être un lieu (une voiture roulant sur la voie de gauche est une anomalie en France, pas au Royaume-Uni) ou un moment (Il est normal d'observer la présence de clients dans un magasin pendant les heures d'ouverture seulement). Les propriétés comportementales permettent de décrire le point en dehors de tout contexte : la voie sur laquelle le véhicule roule, le nombre de clients présents...

Les anomalies collectives concernent des groupes de points qui individuellement

ne présentent pas d'anomalie, mais leur occurrence commune est anormale. Un exemple d'extrasystole est illustré en Figure 2.20.

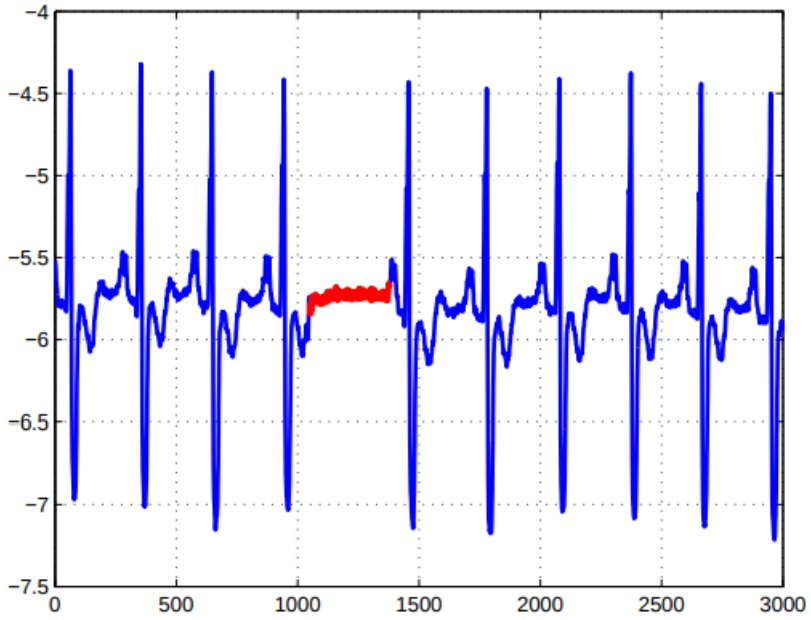

FIGURE 2.20 – Figure extraite de Chandola *et al.* [2009]. Anomalie collective détectée dans le tracé d'un électrocardiogramme humain. Une extrasystole est détectée. La valeur rouge est observée pendant un temps anormalement long, bien qu'elle ne soit pas elle même anormale, témoignant d'une arythmie cardiaque.

2.4.2.1 Applications

La détection d'anomalies est utilisée en sécurité informatique pour détecter des intrusions. Ce domaine d'application est particulier car les données doivent être traitées au fur et à mesure de leur production. Elles sont aussi potentiellement produites en grand volumes. Ce volume rend la détection extrêmement sensible aux faux positifs. En effet, même si une petite proportion de l'activité normale est détectée comme anormale, le nombre brut d'anomalies à traiter risque d'être trop grand. Cette application fait l'objet de plusieurs états de l'art, tels que Butun *et al.* [2014] ou Buczak et Guven [2016].

On utilise aussi la détection d'anomalies pour détecter les fraudes, souvent dans le domaine bancaire [Rieke *et al.*, 2013], ou assurantiel. On s'attache souvent dans cette application à dresser un comportement typique pour chaque client. Une anomalie sera un comportement atypique d'un client par rapport à son propre comportement habituel. Zojaji *et al.* [2016] établissent un état de l'art de la détection d'emploi frauduleux de cartes bancaires. L'état de l'art de Ahmed *et al.* [2016] concerne la

fraude bancaire. Un état de l'art global de la détection de fraude est fourni par Abdallah *et al.* [2016].

Le domaine médical profite également des avancées en détection d'anomalies. Dans ce domaine, un faux négatif peut coûter une vie, on s'attache donc en priorité à ne pas considérer un patient malade en tant que patient sain. Riboni *et al.* [2015] utilisent par exemple une détection de comportement anormal pour diagnostiquer au plus tôt des troubles cognitifs.

La détection de pannes est le domaine dans lequel nous appliquons la détection d'anomalies. Nous nous intéressons à des machines fortement utilisées qui vont donc connaître une usure rapide. Nous disposons généralement de capteurs permettant d'obtenir des informations sur l'état de la machine à intervalles réguliers. Notre étude porte sur la détection de vélos défectueux au sein de systèmes de vélos en libre service. Elle est détaillée en Section 2.5.

Il existe encore bien d'autres domaines d'applications tels que l'analyse d'images, notamment satellites [Li *et al.*, 2015a; Canty, 2014] ; l'analyse de texte avec la détection de nouvelles thématiques [Shin *et al.*, 2017] ; l'apprentissage dirigé par la curiosité [Burda *et al.*, 2018] qui est une forme d'apprentissage automatique reposant sur la détection d'éléments inconnus ; ou encore la détection de perturbations au sein d'écosystèmes [Kou *et al.*, 2006], etc.

2.4.2.2 Approches

Notre application sur la détection de pannes dans les vélos en libre service (Section 2.5) est une détection d'anomalies ponctuelles, c'est ce que nous étudions ici. On ne s'intéresse pas aux approches supervisées qui nécessitent un grand nombre d'exemples. En effet, les anomalies sont la plupart du temps rares face aux exemples normaux et il faut donc regrouper beaucoup de données pour obtenir un nombre d'anomalies suffisant à l'apprentissage supervisé. Les deux principales méthodes de détection d'anomalies non supervisées reposent soit sur le voisinage des anomalies, soit sur du partitionnement.

Comme pour le partitionnement des données, la détection d'anomalies peut se baser sur la densité des exemples. On présume ici qu'un point normal a des caractéristiques proches des autres points normaux, ce qui n'est pas le cas d'un point anormal. La proximité des caractéristiques peut être calculée comme une distance (par exemple euclidienne) ou un indice de similarité. Les approches les plus répandues se classent grossièrement en deux catégories. La première est caractérisée par l'utilisation de méthodes proches de la méthode des k plus proches voisins [Cover et Hart, 1967]. Le score d'anomalie peut être calculé de multiples façons, comme la distance du k-ième plus proche voisin, la somme des distances des k plus proches voisins, le nombre de voisins dont la distance est inférieure à un seuil donné, etc. La deuxième catégorie regroupe les méthodes basées sur l'estimation de la densité au voisinage de chaque exemple [Schubert *et al.*, 2014]. Pour éviter une sensibilité à des régions de densité variable, on compare la densité locale d'un point à la densité locale de ses voisins. Cette méthode est appelée dans la littérature *Local Outlier Factor* [Breunig *et al.*, 2000] (LOF). L'avantage de ces techniques et qu'elles n'ont besoin d'aucun à priori sur la distribution des données. Il suffit de déterminer une

mesure de la distance pour les appliquer à n'importe quel problème. Par contre la mise en production nécessite de calculer la distance aux plus proches voisins pour chaque nouvel exemple, ce qui peut ne pas être efficace.

On peut également appliquer directement des techniques de partitionnement pour détecter des anomalies. On présume ici qu'un point anormal n'appartiendra à aucune partition. Cela ne fonctionne que pour les algorithmes n'attribuant pas obligatoirement une partition à chaque point, tels que DBSCAN [Ester *et al.*, 1996] ou SNN [Ertoz *et al.*, 2002]. Si l'algorithme de partitionnement choisi attribue une partition à tout élément, alors on présume que le point est une anomalie au sein de cette partition, caractérisée par sa distance au centre de la partition. Les algorithmes de partitionnement couramment employés pour ce type de détection sont les algorithmes K-Means, comme nous le faisons en Section 2.5, de cartes auto adaptatives [Kohonen, 1997] (SOM pour l'anglais *Self Organizing Maps*) ainsi que de maximisation d'espérance [Moon, 1996] (EM). Aucune de ces deux présomptions n'est fonctionnelle si l'algorithme de partition crée des partitions de points normaux et des partitions d'anomalies. Dans ce dernier cas, on étudie la taille et la densité des partitions crées pour déceler les anomalies. Une des méthodes utilisées dans ce cas là est nommée FindCBLOF [He *et al.*, 2003]. Elle permet d'assigner un score d'anomalie à chaque partition basé sur la méthode LOF. Il est souvent plus rapide d'appliquer ces méthodes que les méthodes de voisinage car le nombre de partitions est souvent réduit.

2.4.3 Utilisation des Réseaux de neurones

Bien que nous n'ayons utilisé les réseaux de neurones que dans un contexte d'apprentissage supervisé, il existe des applications non supervisées, incluant le partitionnement et la détection d'anomalies. La méthode de partitionnement SOM abordée dans la section précédente se base sur des réseaux de neurones. Son principal attrait repose dans sa capacité à réduire la dimensionnalité des données. C'est aussi le rôle des auto-encodeurs [Hinton et Zemel, 1994]. Ces réseaux de neurones sont entraînés à reproduire leur entrée. Ils sont constitués d'un encodeur, succession de couches comportant de moins en moins de neurones, dont la sortie est une approximation de l'entrée en basse dimension, et d'un décodeur, succession de couches comportant de plus en plus de neurones, permettant de retrouver l'entrée de haute dimension. La structure schématique d'un auto-encodeur est illustrée en Figure 2.21.

Les réseaux antagonistes génératifs [Goodfellow *et al.*, 2014] (GANs pour l'anglais *Generative Adversarial Networks*) représentent un domaine de recherche extrêmement actif concernant l'apprentissage non supervisé à l'aide de réseaux de neurones. Un GAN est composé de deux réseaux de neurones. Le premier est le générateur. Un bruit aléatoire lui est donné en entrée, à partir du quel il doit générer une image. Le second réseau est le discriminateur. Il doit reconnaître si l'image qui lui est donnée en entrée est une image réelle ou si elle a été générée par le générateur. Le générateur essaye de minimiser sa fonction d'erreur $J^{(G)}(\theta_G, \theta_D)$ en modifiant θ_G, il n'a aucun pouvoir sur θ_D. Le discriminateur essaye de minimiser sa fonction d'erreur

5. https://commons.wikimedia.org/wiki/File:Autoencoder_structure.png

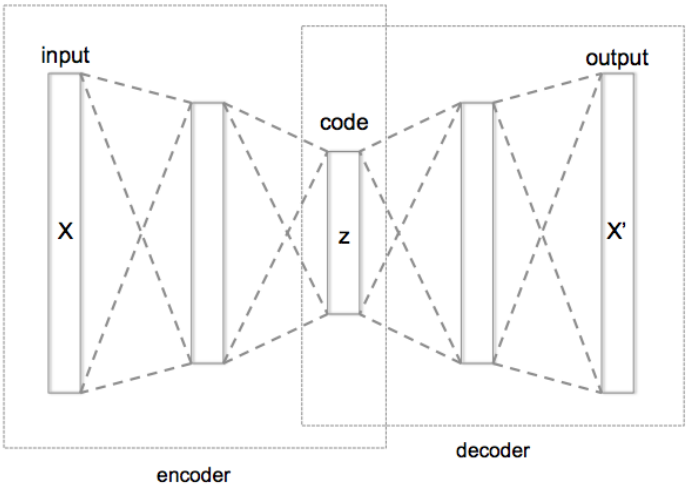

FIGURE 2.21 – Structure schématique d'un auto-encodeur, figure extraite de wikimedia [5]. L'entrée x est codée par l'encodeur pour obtenir une approximation z dans un espace de plus basse dimension. Le décodeur transforme z pour obtenir un élément x' dans le même espace de dimensions que x de manière à en être le plus proche possible.

$J^{(D)}(\theta_G, \theta_D)$ en modifiant θ_D, il n'a aucun pouvoir sur θ_G. Ces deux fonctions sont antagonistes, on considère que le modèle est entraîné quand les deux réseaux ont atteint un équilibre de Nash [Nash, 1951]. Cette méthode n'ayant pas été utilisée au sein de cette thèse, nous ne détaillerons pas plus précisément les mécanismes d'entraînement ou les architectures des GANs. Goodfellow [2016] décrit formellement le processus d'apprentissage des GANs, les fonctions de coût utilisées, ainsi que des techniques pour obtenir de meilleurs résultats.

Ces modèles génératifs sont utilisés par exemple pour créer de nouvelles images, telles que des portraits de personnes imaginaires photo réalistes [Karras *et al.*, 2017], la création de nouveaux personnages d'anime [Jin *et al.*, 2017], l'illustration de texte [Reed *et al.*, 2016]... Ils sont également utilisés pour l'édition d'image, comme l'augmentation de la résolution [Bin *et al.*, 2017] ou le remplissage de trous [Li *et al.*, 2017a], le transfert de style [Luan *et al.*, 2017]... Cette liste n'est pas exhaustive car les applications sont extrêmement nombreuses et l'engouement de la communauté scientifique est fort.

2.5 Application de l'apprentissage non supervisé au problème de détection de vélos défectueux au sein d'un système de vélo en libre service.

Comme nous l'avons vu en Section 1.1.1, les vélos en libre service défectueux posent un problème aux utilisateurs des applications de Qucit, tant particuliers désirant emprunter un vélo que professionnels désirant réparer les pannes au plus vite. Nous proposons ici d'appliquer les méthodes d'apprentissage non supervisé abordées précédemment afin de détecter les vélos défectueux au sein du système de vélos en libre service de New York.

Nous disposons des données décrivant l'ensemble des trajets effectués au sein de ce système, caractérisés principalement par l'identifiant du vélo utilisé, la station et l'heure de départ/arrivée. Nous utilisons ces données pour déterminer pour chaque vélo, jour par jour, s'il était utilisable ou défectueux.

La présente section est organisé comme suit. Nous présentons d'abord les travaux existants sur la détection des anomalies appliqués aux systèmes de vélos en libre-service. Nous décrivons ensuite la méthode proposée, y compris les modules d'extraction de caractéristiques et de détection des anomalies puis le protocole expérimental, notamment la sélection des paramètres et l'infrastructure utilisée. Enfin, nous détaillons les résultats de notre méthode et ses limites.

2.5.1 État de l'art

Bien que les systèmes de vélos en libre-service aient fait l'objet de nombreuses études [Fishman, 2016], peu portent sur la détection des anomalies. Dans un exemple d'une telle proposition Kaspi *et al.* [2015b] utilisent les mêmes trajets que nous utilisons pour simuler le comportement d'une station. Cette étude est donc centrée sur la station. Un modèle stochastique est ensuite décrit. Ce modèle est paramétré par la probabilité de défaillance des vélos qui doit être connue *a priori*. Il est supposé que les utilisateurs ne louent pas de vélos cassés, ce qui n'est vrai que s'il existe d'autres possibilités. Ce modèle n'a pas été confronté à la réalité, de sorte que nous ne pouvons pas savoir si les hypothèses qui ont servi à l'élaboration du modèle étaient préjudiciables.

Bertens *et al.* [2016] ont défini une anomalie de co-occurrence comme deux événements normaux se produisant simultanément quand ils sont habituellement exclusifs. Ils ont essayé de détecter de telles anomalies dans un système de vélos en libre-service. Ils ont utilisé les données de location ainsi que les données météorologiques. Malheureusement, la seule anomalie de co-occurrence qu'ils ont trouvée était une très rare co-occurrence de basse température réelle et de température perçue élevée.

Pour autant que nous le sachions, aucun algorithme d'apprentissage automatique n'a été appliqué aux systèmes de vélos en libre-service dans le but de détecter les vélos cassés, ce qui est l'objectif de notre travail. Ce manque d'études s'explique en partie par le fait qu'aucun étiquetage n'est disponible pour entraîner les modèles d'apprentissage automatique. Aucun opérateur ne partage des données d'entretien détaillées, telles que les dates et la nature des réparations de chaque vélo.

Cependant, certains opérateurs, comme Motivate, partagent le nombre de réparations qui ont lieu chaque mois. Musicant *et al.* [2007] décrit des modèles d'apprentissage automatique qui peuvent utiliser un tel résultat agrégé au lieu des étiquettes usuelles pour être entraînés .

Aussi, Fanaee-T et Gama [2014] expliquent comment sans étiquette préalable, ils étaient capables d'étiqueter les événements avec des données de partage de vélo. Mais les événements étiquetés n'ont pas eu lieu dans le système de vélos en libre-service. Ce dernier est plutôt utilisé comme détecteur d'événements et les connaissances de base extraites de Google sont utilisées pour fournir des étiquettes.

2.5.2 Méthode proposée

Afin de déterminer pour chaque vélo, jour par jour, s'il était utilisable ou défectueux, nous proposons une méthode qui peut être appliquée en deux étapes indépendantes. La première étape consiste à extraire les caractéristiques des données brutes, tandis que la suivante utilise ces données améliorées et remaniées pour détecter les anomalies. Un diagramme récapitulant le processus est présenté dans la Figure 2.22.

2.5.2.1 Extraction des caractéristiques

L'étape d'extraction des caractéristiques est fortement liée à la source d'information disponible. À ce jour, seul Motivate propose des données qui pourraient être utilisées pour la détection de pannes. Ces données sont une agrégation des étiquettes souhaitées, ce qui nous amène à un problème de classification des sorties agrégées.

Nous disposons de deux types de données.

(i) Le premier est massif et précis ; il décrit les trajets effectués avec le système de vélos en libre-service CitiBike qui peut être téléchargé sur le site Web de CitiBike [Motivate, 2016b]. Chaque trajet est caractérisé par des données hétérogènes décrites dans le tableau 2.1. Nous appelons cette table l'ensemble de données des déplacements.

(ii) Le second est plutôt petit et imprécis. Il consiste en un ensemble de rapports mensuels [Motivate, 2016a] décrivant les activités de l'opérateur et des utilisateurs pendant le mois. Il fournit diverses informations telles que le nombre moyen de trajets quotidiens par vélo, le nombre d'utilisateurs, le respect de l'accord de niveau de service et, celui que nous utilisons pour optimiser notre modèle, le nombre de réparations de vélos. Nous appelons cet ensemble de données la vérité terrain. Notons que nous ne savons pas ce qui se passe lorsqu'un vélo a été réparé deux fois dans le mois (est-ce décompté comme une ou deux réparations de vélo ?). Nous ne savons pas à quel point la casse doit être grave pour que le vélo soit pris en charge par l'équipe de maintenance.

Nous utilisons l'ensemble de données des trajets pour construire un ensemble de caractéristiques, utilisé conjointement avec l'ensemble de données de vérité terrain pour entraîner et évaluer notre modèle.

Cette étude n'est pas centrée sur les trajets, de sorte qu'elle ne caractérise pas un trajet pour savoir s'il s'agit ou non d'un trajet de vélo cassé. En effet, l'absence de trajet peut être lui même un signe d'anomalie. Au lieu de cela, nous construisons un

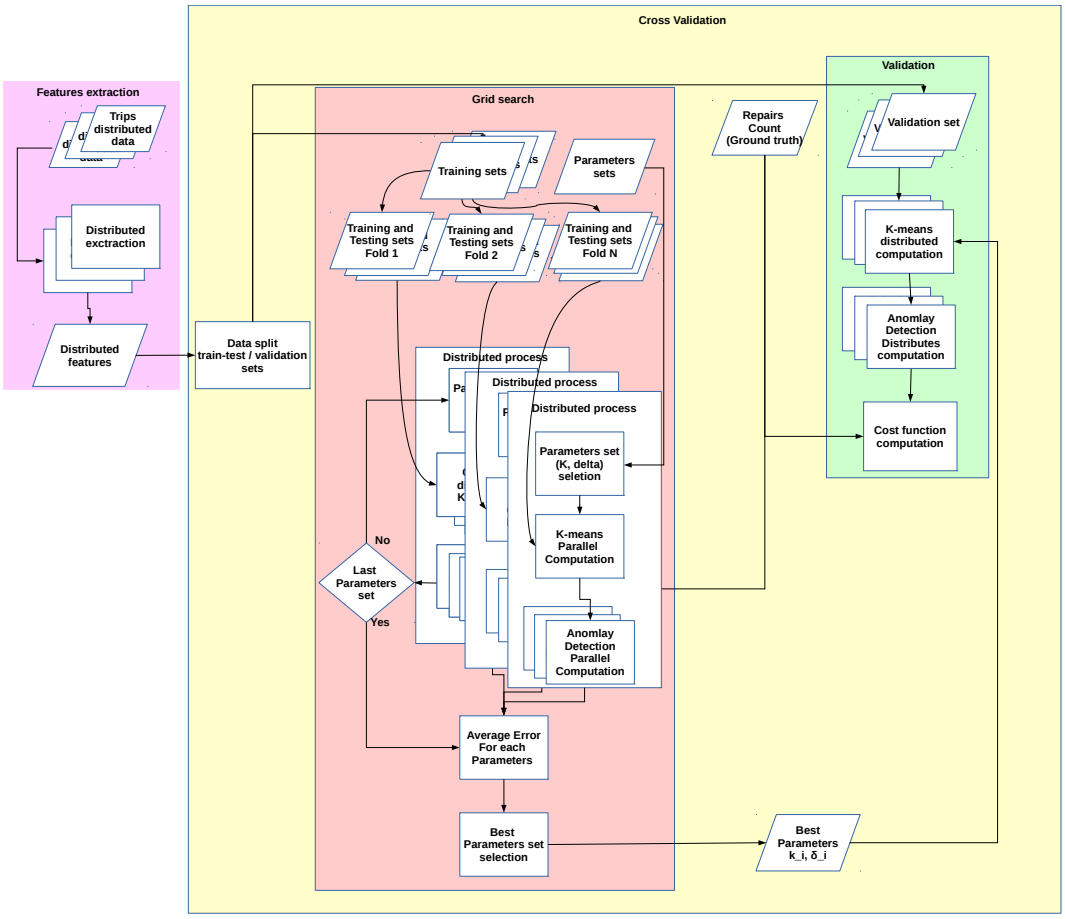

FIGURE 2.22 – **Aperçu du processus.** Cette figure présente les deux principaux processus distincts. Tout d'abord, l'extraction des caractéristiques crée un ensemble de caractéristiques à partir des données brutes. Ensuite, le processus de validation croisée qui entraîne et évalue le modèle. Lors de la recherche par grille, une erreur est calculée pour chaque configuration de paramètres et pour chaque échantillon Une erreur moyenne sur les échantillons est ensuite calculée pour chaque ensemble de paramètres et celui avec l'erreur la plus faible est sélectionné. Il est ensuite utilisé pour effectuer une classification sur l'ensemble de test qui sont des caractéristiques qui n'ont jamais été utilisées pour entraîner le modèle. L'erreur obtenue sur l'ensemble de test est un indicateur de la façon dont notre modèle peut généraliser aux nouvelles données.

.

échantillon par jour et par vélo, puis nous déterminons si, à un jour donné, un vélo donné était défectueux ou non. Pour chacun de ces échantillons, les caractéristiques personnalisées sont des statistiques sur l'utilisation du vélo pendant un intervalle donné de T jours. Ils caractérisent le dernier de ces jours. En glissant une fenêtre de T jours pour chaque vélo, plusieurs millions d'échantillons sont calculés. Si pendant la durée d'une fenêtre (de période T) un vélo donné n'a pas été utilisé du tout, alors

TABLE 2.1 – Description des données brutes

Caractéristique	Description	Fréquence
Lieu de départ	Coordonnées	Toujours
Lieu d'arrivée	Coordonnées	Toujours
Station de départ	Nom de station	Toujours
Station d'arrivée	Nom de station	Toujours
Date de départ	Horodatage	Toujours
Date d'arrivée	Horodatage	Toujours
Vélo	Identifiant unique	Toujours
Naissance	Date de naissance de l'utilsateur	Si abonné
Genre	Genre de l'utilisateur	Si abonné
Réparations	Nombre de réparations rapporté	Mensuel

cet échantillon n'est pas créé.

Les caractéristiques extraites sont les suivantes : nombre de voyages effectués, nombre de stations de départ distinctes, nombre de stations d'arrivée distinctes, nombre de boucles (même station d'arrivée et de départ), la distance moyenne parcourue au cours des trajets, la distance maximale parcourue au cours des trajets, la distance minimale parcourue au cours des trajets, la durée moyenne des trajets, la durée maximale des trajets, durée minimale des trajets.

Cela signifie que chaque échantillon existe dans un espace de 10 dimensions, chacune de ces caractéristiques étant une de ces dimensions.

Ces caractéristiques sont choisies en fonction de trois critères. Tout d'abord, il s'agit d'un moyen facile d'agréger plusieurs jours de trajets pour un vélo. Deuxièmement, une fois la classification effectuée, il est possible de définir des heuristiques pour en vérifier la cohérence. Par exemple, un vélo cassé n'effectuerait pas beaucoup de longs trajets à travers la ville par rapport aux autres vélos. Enfin, la forme de la distribution de chaque entité est soit gaussienne, soit au moins avec une longue queue d'événements aberrants, voir Figure 2.23. Cette forme est souhaitable pour la détection d'anomalies car elle permet de séparer facilement les événements rares des événements normaux.

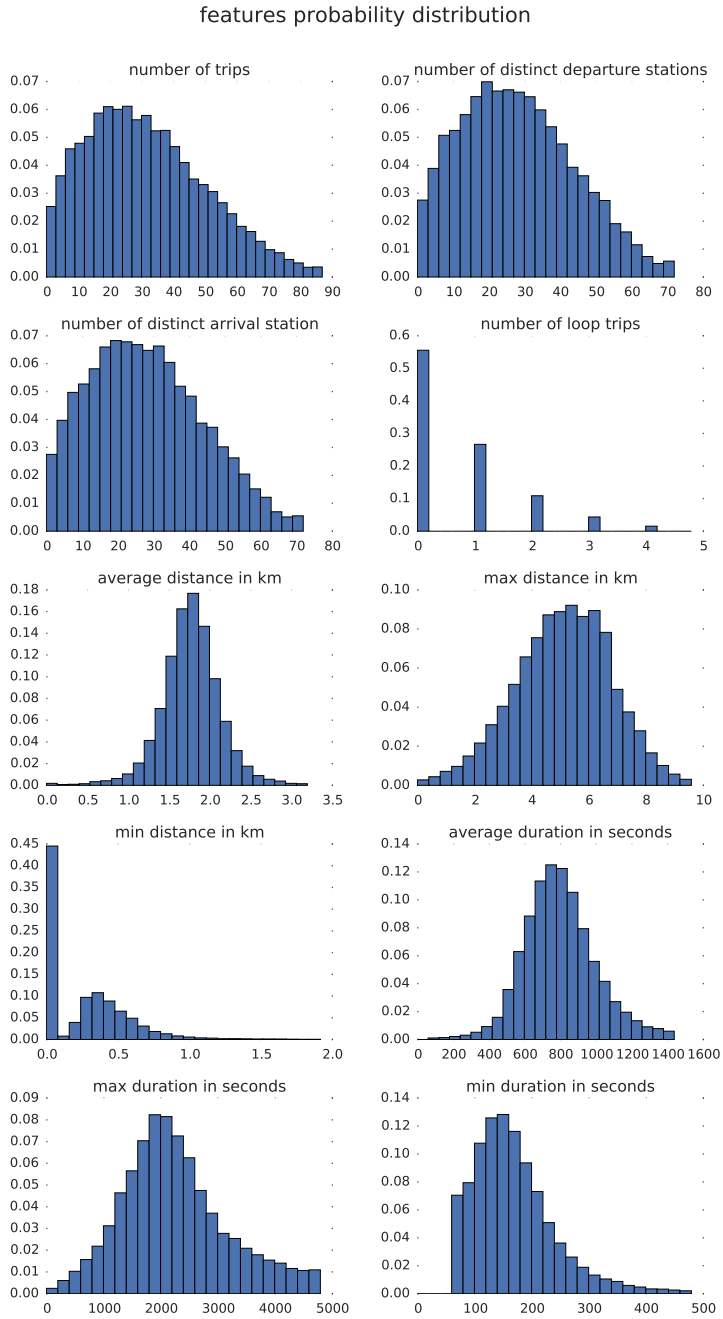

FIGURE 2.23 – Distribution de probabilités de toutes les caractéristiques extraites.

2.5.2.2 Détection d'anomalies

Après l'extraction des caractéristiques de chaque couple de jour et de vélo, il est nécessaire de les classer dans l'une des deux catégories : normal si le vélo fonctionne bien, anormal si le vélo doit être ramassé par l'équipe de maintenance.

Musicant *et al.* [2007] proposent des algorithmes pour effectuer un apprentissage automatique sur des sorties agrégées comme celles-ci. Cependant, il existe une différence majeure entre nos données et les leurs qui rend leur solution non directement applicable. Dans leur cas, la sortie agrégée est la somme des sorties qui seraient normalement utilisées, ce qui n'est pas notre cas.

Voici un exemple de problème dont la sortie agrégée est la somme des sorties. Soit un ensemble de cinq boules ; définissez un modèle classant chaque boule comme une boule bleue ou une boule rouge. La vérité terrain indique qu'il y a 3 boules bleues et 2 boules rouges. La sortie du modèle est une classification boule par boule, la sortie agrégée une somme sur chaque classe.

Dans nos travaux, la sortie du modèle est pour chaque vélo et chaque jour une classification entre les classes «fonctionnel» et «défectueux». Une sortie agrégée correspondante serait la somme sur un mois du nombre de vélos défectueux jour par jour. Cependant, la sortie agrégée dont nous disposons est la somme des réparations qui ont eu lieu au cours de chaque mois. Il y aura généralement plusieurs jours durant lesquels un vélo est défectueux avant d'être réparé, ces deux agrégations sont donc différentes. Leur méthode serait pertinente si l'on essayait de classer chaque jour de la vie d'un vélo comme étant un jour où une réparation a eu lieu.

Puisqu'il n'y a pas d'information sur la possibilité qu'un vélo soit réparé plusieurs fois au cours d'un mois ou sur l'impact que cela aurait sur le nombre de réparations rapportées, on suppose ici qu'un vélo n'est réparé qu'une fois par mois au maximum. La fonction d'agrégation utilisée est donc un comptage des vélos distincts qui ont été classés comme cassés chaque mois. En conséquence, les algorithmes proposés par Musicant *et al.* ne peuvent pas être utilisés.

La méthodologie de classification se fait en deux étapes. (i) La première étape consiste à calculer un partitionnement qui sépare les caractéristiques de différents groupes représentant différents comportements normaux d'un vélo, (ii) puis la deuxième étape consiste à identifier les échantillons qui n'appartiennent pas clairement à une partition. Ce sont eux qui sont trop éloignés du centre de partition le plus proche.

Tout d'abord, nous supposons qu'un vélo a un comportement différent en fonction de sa situation. Par exemple, un vélo amarré près d'un espace vert comme un parc ou un plan d'eau n'est pas utilisé de la même façon qu'un vélo stationné près d'un quartier résidentiel. Le premier est plus susceptible d'être utilisé pour une promenade, se terminant dans la même station qu'au départ. Le second peut être utilisé tôt le matin pour se rendre dans une zone de travail et le soir, pour retourner dans une zone d'habitation.

Ceci nous amène à effectuer un partitionnement sur nos échantillons pour séparer les différents comportements normaux. Pour faire ce partitionnement, nous utilisons l'algorithme K-Means [MacQueen, 1967]. K, le nombre de partitions et donc le nombre de comportements normaux pertinents, étant un hyper-paramètre de notre modèle. Chaque échantillon appartient à une partition k, avec des parti-

tions indexées par $k \in [1; K]$ de taille M^k. Chaque échantillon d'une partition k est désigné par $S_i^k, i \in [1; M^k]$. Chaque partition possède un centroïde, C^k, qui est la position moyenne de tous les points de la partition k. On peut placer un centroïde dans notre espace multidimensionnel en définissant chacune de ses caractéristiques $\{C_j^k\}, j \in [1; n]$ comme moyenne des valeurs de cette caractéristique pour les échantillons appartenant à cette partition ; $C_j^k = \frac{\sum_{i=1}^{M^k} S_{ij}^k}{M^k}$.

Une fois que les échantillons ont été assignés à une partition, nous devons encore déterminer lesquels ne sont pas normaux. Puisque nos partitions sont supposées représenter des comportements normaux différents, nous supposons que les comportements anormaux sont ceux qui ne sont pas clairement dans une partition. Nous considérons qu'un échantillon n'est pas clairement dans une partition lorsque sa distance au centroïde est significativement plus grande que celle des autres membres de la partition à ce même centroïde.

Cette distance est définie comme suit :

$$dist\left(S_i^k, C^k\right) = \sqrt{\sum_{j=1}^{n}\left(S_{ij}^k - C_j^k\right)^2} \tag{2.38}$$

Ensuite, nous pouvons définir la fonction qui est utilisée pour étiqueter l'échantillon comme normal ou anormal :

$$anom(S_i) = \begin{cases} 1 & \text{si} \quad dist\left(S_i^k, C^k\right) > \delta * \sigma^k \\ 0 & \text{sinon} \end{cases} \tag{2.39}$$

où *anom* est une fonction qui retourne 1 si l'échantillon S_i est anormal (le vélo est censé être défectueux) et 0 s'il est normal (le vélo n'est pas défectueux), σ^k est l'écart type des distances au centre, δ est un autre paramètre du modèle, utilisé pour moduler la distance maximale au centroïde à partir de laquelle un échantillon est considéré anormal.

2.5.3 Protocole Expérimental

Nous décrivons ici le protocole suivi pour entraîner et évaluer notre modèle.

Il reste deux hyper-paramètres à optimiser : K le nombre de partitions (*i.e.*, le nombre de comportements pertinents) et δ le facteur de l'écart-type de la distance au centroïde à partir de laquelle un échantillon est classé comme une anomalie. T, le nombre de jours utilisés pour construire les fonctionnalités n'est pas optimisé ici mais fixé à sept.

L'optimisation des hyper-paramètres est contrôlée par un processus de validation croisée. On utilise un an et demi de données, de juin 2013 à décembre 2014. Ensuite, une fois les paramètres fixés, nous effectuons une classification sur une période de 15 mois, de janvier 2015 à mars 2016, qui est notre ensemble de test. Au cours de la phase de sélection, l'ensemble de données des caractéristiques a été séparé cinq fois dans un ensemble d'apprentissage et un ensemble de validation (validation

croisée à 5 échantillons). Chaque fois, tous les mois de juin 2013 à décembre 2014 sont utilisés, 80% d'entre eux pour l'apprentissage et 20% pour calculer l'erreur (le résultat de la fonction de coût décrite dans l'équation 2.40) sur l'ensemble de validation. Nous utilisons la racine de l'erreur quadratique moyenne (RMSE). L'erreur d'un couple d'hyper-paramètres est l'erreur de validation moyenne des cinq différents échantillons. Ensuite, on sélectionne le jeu de paramètres ayant la plus petite erreur moyenne sur l'ensemble de validation et on exécute la classification sur l'ensemble de test.

Nous trouvons le meilleur ensemble d'hyper-paramètres grâce à une recherche par grille puisque l'espace de recherche est assez restreint pour effectuer une recherche exhaustive. K est recherché entre 1 et 5 ; 1 signifie qu'il n'y a pas besoin d'une étape de partitionnement, 5 est assez élevé pour que plusieurs comportements puissent apparaître et assez bas pour que le temps de calcul soit raisonnable. Si l'erreur avait présenté une tendance à diminuer au fur et à mesure que K augmentait, nous aurions dû pousser cette limite plus loin, mais cela n'a pas été le cas. δ est recherché entre 1 et 3, avec des pas de 0.25. Beaucoup de nos caractéristiques ayant une forme gaussienne, nous savons qu'avec $\sigma < 1$ nous avons trop d'échantillons anormaux et avec $\sigma > 3$ nous en avons trop peu. Afin de juger si un ensemble de paramètres est meilleur qu'un autre, nous définissons une fonction de coût (équation 2.40) que les modèles doivent minimiser. Elle mesure la distance entre le nombre de vélos distincts ayant été classifiés comme anormaux au cours d'un mois et le nombre de réparations signalées pour ce mois.

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(Anomalies_{K,\delta}(i) - Reparations(i)\right)^2} \tag{2.40}$$

avec $Anomalies_{K,\delta}(m)$ le nombre de vélos distincts qui ont été classés comme cassés au moins une fois durant le mois i et N le nombre de mois utilisés.

Les domaines de valeurs des caractéristiques extraites sont différents ce qui pourrait conduire la distance calculée entre deux vecteurs de cet espace multidimensionnel à dépendre presque exclusivement de la différence entre une de leurs dimensions. Il est nécessaire de les normaliser, afin que chaque caractéristique ait la même moyenne et une plage de valeurs similaire.

Cette procédure a été implémentée avec Spark [Zaharia *et al.*, 2010] 1.6.0. Nous avons exécuté l'extraction des caractéristiques et la sélection des paramètres sur une machine de bureau avec un processeur i7 8 cœurs, 32 Go de RAM et un disque dur SSD de 100 Go et un cluster de 15 machines, chacune avec 50 Go de RAM et 48 cœurs, utilisant les données stockées dans Hadoop Distributed File System. La configuration alloue 50 travailleurs avec 5 cœurs par travailleur (il y a plus d'un travailleur par machine) et la possibilité d'utiliser 5 Go de RAM par travailleur.

2.5.4 Résultats

2.5.4.1 Performances

Cette section présente les résultats obtenus avec l'ensemble optimal de paramètres, *i.e.* $K = 1$ et $\delta = 2.25$ comme illustré dans la Figure 2.24. Ces paramètres nous fournissent un modèle plus performant avec les caractéristiques choisies et la fonction de coût sur des données non vues par le modèle lors de l'apprentissage.

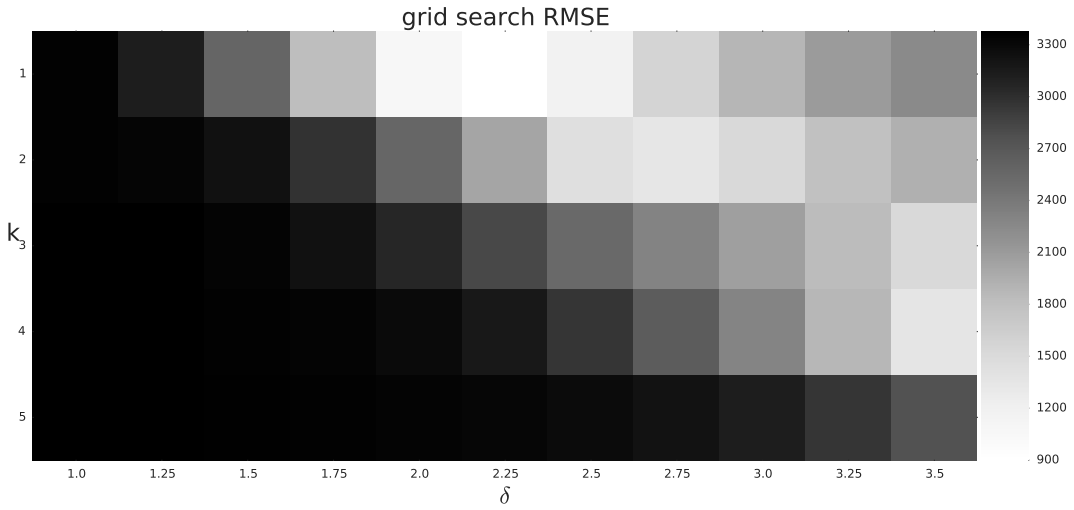

FIGURE 2.24 – **Meilleurs hyper-paramètres** Ceci représente l'erreur moyenne sur les 5 ensembles de validation avec chaque combinaison d'hyper-paramètres. Plus la couleur est claire, plus l'erreur est faible. L'erreur minimale est atteinte à $K = 1$ et $\delta = 2.25$.

L'équation 2.40 implique que nous essayons de minimiser la distance entre deux valeurs ; le nombre donné de réparations pour chaque mois et le nombre de vélos distincts qui ont été détectés comme défectueux au cours du dit mois au moins une fois. La figure 2.25 présente l'évolution de ces valeurs. La période d'apprentissage sur laquelle nous avons effectué la validation croisée prend fin en décembre 2014. La période de test débute en janvier 2015. Au cours de cette dernière, nous effectuons une classification sur des données que nous n'avons jamais vues auparavant. En utilisant la racine de l'erreur quadratique moyenne (RMSE), nous calculons une erreur qui dépend de la distance entre les deux courbes.

Sur l'ensemble d'apprentissage, le RMSE est de 852 et sur l'ensemble de validation 1 304, indiquant un sur-apprentissage potentiel. Les prédictions sur l'année 2016 s'éloignent significativement du nombre de réparations rapportées. Ce dernier atteint des valeurs record, peut être témoin d'un changement de politique de réparation.

Nous avons calculé une corrélation croisée normalisée (Équation 2.41) entre les deux courbes pendant la période de formation et la période de validation. Plus elle est proche de 1, plus notre classification et la vérité de terrain suivent les mêmes variations. La corrélation croisée normalisée est définie comme suit :

$$\frac{1}{n}\sum_{x}\frac{\left(f(x)-\overline{f}\right)\left(t(x)-\overline{t}\right)}{\sigma_f\sigma_t} \tag{2.41}$$

où $t(x)$ est la valeur à l'indice x dans le vecteur contenant les valeurs du nombre de vélos distincts classifiés comme défectueux mois par mois, $f(x)$ est la valeur à l'indice x dans le vecteur contenant les valeurs du nombre de réparations déclarées chaque mois, n est le nombre de mois dans $t(x)$ et $f(x)$, $\overline{f}$ est la moyenne de f et σ_f est l'écart type de f.

La corrélation croisée normalisée pendant la période d'entraînement est de 0,47 et pendant la période de validation de 0,35. Le fait d'avoir des valeurs faibles à la fois pour la RMSE et pour les mesures de corrélation indique que le δ sélectionné est utile pour ajuster le nombre moyen de réparations, mais ne nous permet pas de prévoir la variabilité.

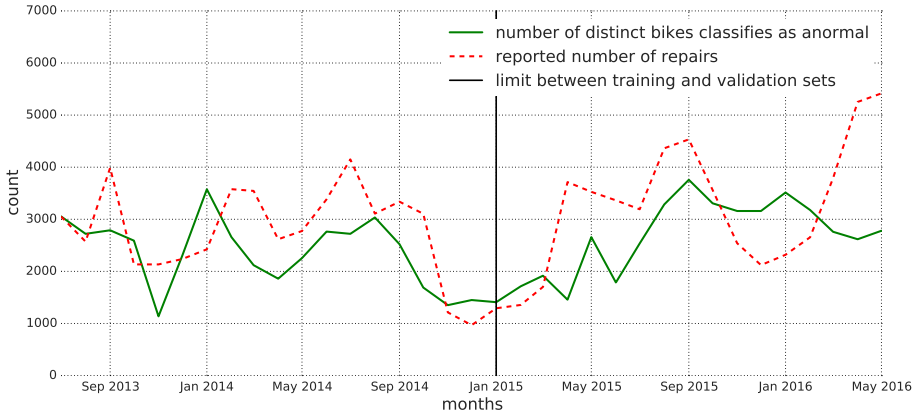

FIGURE 2.25 – **Classification VS vérité terrain.** La courbe en pointillés rouges indique le nombre de réparations qui ont été signalées chaque mois. La courbe verte représente le nombre de vélos distincts qui ont été classifiés comme anormaux au moins une fois ce mois-ci. La période d'apprentissage s'étend du début jusqu'en décembre 2014. L'ensemble de validation commence en janvier 2015. Ils sont séparés par une ligne verticale noire.

L'implémentation sur une architecture cloud est justifiée par la taille des données. Sur une machine classique, l'extraction des fonctionnalités nécessite 25 minutes de traitement. L'optimisation des paramètres se fait en 2 heures sans validation croisée et ne peut être réalisée avec une validation croisée. Même exécutée avec une implémentation Spark parallèle, le disque est plein et le processus s'arrête après plusieurs heures de calcul. Sur le cluster, l'extraction des fonctionnalités s'effectue en 2 minutes et 30 secondes, l'optimisation sans validation croisée en 17 minutes et avec la validation croisée en 2 heures.

2.5.4.2 Discussion

Puisque $K = 1$ est le meilleur paramètre, nous n'avons pas réussi à identifier plusieurs comportements normaux. Trois explications sont possibles.

Premièrement, il n'est peut-être pas optimal d'essayer de séparer ces comportements et aucune solution avec $k > 1$ ne peut être meilleure.

Deuxièmement, l'explication la plus probable est que les caractéristiques choisies ne permettent pas de séparer les comportements. En effet, nous n'avons pas été en mesure de visualiser une séparation entre les clusters obtenus avec l'algorithme de K-Means : les visualisations de nuage de points sur chaque paire de dimensions ne les séparent pas. Nous devons trouver d'autres caractéristiques qui mèneront à un regroupement significatif. Nous pourrions utiliser t-sne [Maaten et Hinton, 2008] ou d'autres techniques avancées de visualisation de données de haute dimension afin de les distinguer. Si ce n'est pas possible, nous formerons un modèle linéaire sur nos caractéristiques avec notre classification comme cible afin d'analyser les paramètres d'un tel modèle. Nous aurons ensuite un aperçu de ce qui rend une caractéristique plus susceptible de se séparer en grappes.

La troisième possibilité est que la fonction de coût ne favorise pas une telle séparation. Plusieurs indices indiquent que cette fonction est actuellement sous-optimale. Par exemple, s'il n'est pas inhabituel de voir un vélo réparé plusieurs fois par mois, et si nous sommes sûrs que chaque vélo cassé est réparé au cours du mois, alors le nombre distinct de vélos cassés que notre modèle produit devrait toujours être inférieur au nombre enregistré de réparations. Nous pénaliserons donc lourdement le cas contraire dans notre fonction de coût. D'autre part, on s'attend à ce que la corrélation soit élevée, il faut donc la favoriser au sein de la fonction de coût. Enfin, le fait qu'un vélo ne soit détecté comme cassé qu'une seule journée ou tous les jours du mois ne fait aucune différence. Un terme de régularisation aidera à maintenir le nombre global d'anomalies à un niveau inférieur.

Une autre limite de notre approche est que nous ne pouvons pas encore déterminer si les anomalies détectées correspondent à des vélos cassés. La courbe qui représente la performance de notre classement (courbe verte) peut être très proche de la courbe représentant le nombre réel de réparations (courbe rouge), sans représenter les vélos cassés. Elle pourrait par exemple représenter le nombre de réparations.

Nous pourrions dans un premier temps utiliser un algorithme classique de classification à sorties agrégées pour essayer de détecter si chaque vélo a été pris par l'équipe de maintenance pour une réparation, plutôt que de détecter un vélo défectueux. Le nombre de ces occurrences est supposé correspondre aux étiquettes agrégées données. Ainsi, des algorithmes de classification à sorties agrégées, conçus pour ce cas précis, auront probablement de meilleures performances. Cela permet également de traiter sans ambiguïté le cas où un vélo est ramassé plus d'une fois. Ensuite, si ce nouveau modèle est assez bon, nous pouvons utiliser ses résultats comme de nouvelles étiquettes non agrégées. Grâce à cette nouvelle information, nous aurions une meilleure façon d'évaluer l'efficacité de notre modèle actuel. En effet, toutes les anomalies que nous détectons doivent se produire peu de temps avant que le vélo ne soit récupéré.

Enfin, le facteur de distance déterminant si un échantillon est trop éloigné de son centroïde pour être classé comme normal, δ, peut être unique pour chaque groupe.

L'avantage serait que nous pourrions avoir des groupes entiers d'anomalies. Pour l'instant, nous ne pouvons pas le permettre, car cela rendrait notre modèle très complexe. Avec autant d'hyper-paramètres à optimiser, le temps de calcul et la probabilité de sur-apprentissage sont très élevés.

2.5.5 Conclusion

Les systèmes de vélo en libre-service font face à des problèmes de vélos défectueux et cette application présente une méthodologie aidant à les détecter. En utilisant les données des trajets CitiBike ainsi que le nombre de réparations mensuelles, nous proposons une méthode pour classer chaque jour de la vie de chaque vélo comme un jour où ce vélo n'est pas utilisé, cassé ou fonctionnel. Pour ce faire, nous extrayons des caractéristiques en calculant des statistiques sur l'utilisation du vélo au cours des jours précédents, puis nous appliquons un algorithme de partitionnement sur les échantillons obtenus. Nous détectons alors les valeurs aberrantes.

Nous constatons qu'avec les caractéristiques conçues et la fonction de coût utilisée, le regroupement n'est pas nécessaire. En classant les échantillons trop éloignés de la moyenne de tous les échantillons comme des vélos cassés, on obtient un bon ajustement à la courbe représentant le nombre de réparations chaque mois. Cependant, une faible corrélation croisée normalisée indique que le nombre de vélos distincts classés au moins une fois comme brisés durant le mois ne suit pas les mêmes variations que le nombre de réparations. C'est un indice du fait que les anomalies détectées ne correspondent pas à des vélos cassés.

Chapitre 3

Détection d'anomalies dans les systèmes de cartographie

Ce chapitre étudie l'application des techniques d'apprentissage automatique supervisé pour la segmentation d'images satellites afin d'en extraire les contours des bâtiments. Ces derniers peuvent être comparés aux données d'OSM pour contrôler leur justesse. Le traitement de ce problème nous a permis de proposer trois contributions. La première, est une analyse des différents types d'étiquettes utilisables pour ce problème. La deuxième contribution, une étude sur la fusion de modèles de segmentation, a fait l'objection d'une publication au sein d'un workshop de CVPR 2018 [Delassus et Giot, 2018]. La dernière est une étude sur la résistance au bruit dans les labels qui fera prochainement l'objet d'une soumission.

On dresse d'abord en Section 3.1 un état de l'art des techniques d'apprentissage profond appliquées à la segmentation d'images. Les architectures les plus récentes sont étudiées, en particulier l'innovation apportée par chacune d'entre elles. Les challenges SpaceNet, qui furent aussi bien la source de nos données que le cadre de nos expérimentations sont présentés en Section 3.2. La Section 3.3 témoigne de nos expérimentations dans le domaine de la détection de bâtiments et ouvre la voie à une étude sur les différents types d'étiquettes pouvant être extraits de la vérité terrain. Notre contribution sur la modélisation ensembliste fait l'objet de la Section 3.4. Nous y comparons trois méthodes de fusion proposées à la méthode de fusion standard dans l'état de l'art. Pour finir, la Section 3.5 présente notre contribution finale, un protocole expérimental permettant d'analyser les performances de modèles en fonction de la qualité de l'étiquetage du jeu d'apprentissage. Des solutions pour renforcer le modèle face à ces perturbations y sont proposées et comparées.

3.1 Prérequis : apprentissage profond pour la segmentation d'image

3.1.1 Analyse d'images pour la segmentation

L'analyse d'une image consiste à donner un sens sémantique à des éléments qu'elle représente. À partir d'une matrice de pixels, il s'agit d'extraire de la connaissance de plus haut niveau.

3.1.1.1 Applications

Grâce à une fiabilité de plus en plus grande, les techniques d'analyse d'image sont utilisées dans différents domaines tels que la télédétection, l'imagerie médicale, la conduite autonome, la reconnaissance faciale [Robertson et Burton, 2016], etc. Dans chaque cas, on cherche à extraire des connaissances de haut niveau depuis des matrices de pixels, données de bas niveau. Dans cette thèse, les connaissances de haut niveau à extraire des images se trouvent sous forme de données géographiques permettant de décrire la topographie urbaine, tels que des bâtiments, des routes, des arbres, etc. Les images médicales, elles, sont utilisées pour l'analyse des résultats d'imagerie par résonance magnétique [Sharma *et al.*, 2017] par exemple, où la connaissance se trouve dans le diagnostique d'une maladie ou le suivi de son évolution. La connaissance à extraire d'une image lors d'une reconnaissance faciale correspond à l'identité de la personne dont le visage est représenté dans l'image.

Effectuer les analyses d'images automatiquement plutôt que manuellement présente plusieurs avantages. Le principal avantage est le passage à l'échelle. Dans notre cas le but est d'analyser toute une ville en quelques minutes, ce qu'un humain n'est pas capable de faire. Un deuxième avantage est la performance. En effet, sur certaines tâches de reconnaissance d'image la machine surpasse l'homme. En 2015, He *et al.* [2015] mettent au point un modèle de reconnaissance d'image ayant une meilleure précision que les humains sur le jeu de données ImageNet [1], battu peu de temps après par Ioffe et Szegedy [2015]. Le troisième avantage est de pouvoir utiliser directement la connaissance extraite des images au sein d'autres applications. Par exemple les véhicules autonomes [Ziegler *et al.*, 2014] utilisent la reconnaissance d'image pour reconnaître leur environnement, et utilisent cette donnée pour se diriger.

3.1.1.2 Principe

Pour reconnaître tous les éléments contenus dans une image satellite, nous essayons de classer chaque pixel de l'image dans une des catégories d'objets à reconnaître prédéfinies. Cette classification au niveau pixel est appelée segmentation sémantique. Un exemple de segmentation est montré en figure 3.1.

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

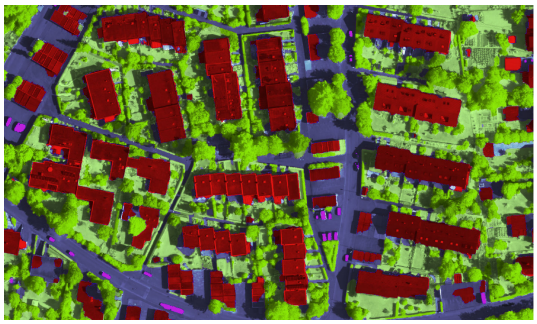

FIGURE 3.1 – Marmanis *et al.* [2016] ont mis au point un modèle de segmentation sémantique d'images satellites. Chaque pixel de l'image a été classifié et est coloré en fonction de sa classe. Les pixels rouges représentent les bâtiments, les verts foncés des arbres, les verts clairs de la végétation basse, le bleu des surfaces imperméables et le fuchsia des voitures.

3.1.2 État de l'art des techniques d'apprentissage profond pour l'analyse d'image

3.1.2.1 Avant l'apprentissage profond

Jusque dans les années 2010, les méthodes d'apprentissage pour la segmentation se basaient principalement sur des techniques de SVM [Song et Civco, 2004] ou de forêts aléatoires [Shotton *et al.*, 2013, 2008], s'appliquant sur des propriétés extraites manuellement des images, telles que la détection d'arêtes ou d'angles, transformation de caractéristiques visuelles invariante à l'échelle [Lowe, 1999], application de seuils, correspondance de templates [Yuille *et al.*, 1992], regroupement de pixels par couleur, contours actifs [Mille *et al.*, 2006], etc. Ces propriétés étaient également utilisées précédemment, alors que les premières techniques de segmentation ne reposaient pas sur l'apprentissage [Brice et Fennema, 1970; Brun, 1996; Cinque *et al.*, 2004; Ma *et al.*, 2009].

Aujourd'hui, plusieurs facteurs nous mènent à l'utilisation de techniques d'apprentissage profond pour de meilleurs résultats. Tout d'abord l'abondance d'images. Les réseaux de neurones artificiels nécessitent un grand nombre d'images durant le processus d'entraînement. Ce besoin est satisfait par les multiples jeux de données de plusieurs millions d'images étiquetées qui sont ouverts à tous[Nech et Kemelmacher-Shlizerman, 2017; Guo *et al.*, 2016; Wang *et al.*, 2018; Russakovsky *et al.*, 2015], et encore plus de jeux de données de plusieurs milliers d'images [2] publiés récemment. Ensuite, les améliorations matérielles ont permis de diminuer les temps d'apprentissage et d'augmenter le nombre d'images pouvant être utilisées. Malgré ces améliorations, certains modèles tels que Mask R-CNN [He *et al.*, 2017] demandent encore plusieurs jours d'apprentissage en utilisant huit cartes graphiques haut de gamme, les rendant difficilement utilisables à moins de disposer de matériel onéreux. Pour finir, l'apprentissage profond est un domaine de recherche très actif connaissant régulièrement des

2. http://deeplearning.net/datasets/

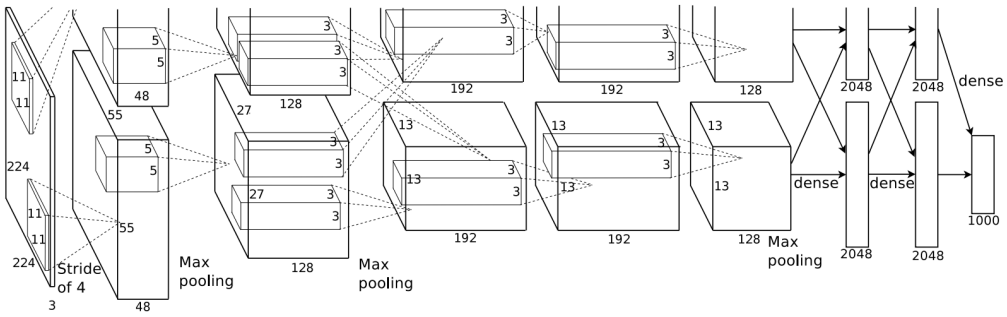

FIGURE 3.2 – Architecture du réseau AlexNet, figure extraite de l'article original.

avancées significatives, comme nous le voyons dans cette section.

3.1.2.2 Modèles de références

Quelques modèles servent souvent de base pour créer de nouvelles architectures, qui soit en seront des variantes, soit les intégreront au sein de leur propre architecture afin de les augmenter. Ce sont les modèles AlexNet [Krizhevsky *et al.*, 2012], VGG-16 [Simonyan et Zisserman, 2014], et ResNet [He *et al.*, 2016]. L'utilisation de ces modèles comme base permet de réutiliser leurs paramètres et ainsi partir d'un modèle pré-entraîné. L'apprentissage est alors accéléré, mais permet également d'atteindre de meilleures performances [Pratt, 1993].

AlexNet [Krizhevsky *et al.*, 2012] est le premier modèle d'apprentissage profond à sortir vainqueur du concours de reconnaissance d'images ImageNet [Russakovsky *et al.*, 2015] en 2012. Ses performances remarquables à l'époque (TOP-5 précision de 84.6% contre 73.8% pour le second) relancent l'intérêt de la communauté scientifique pour ces modèles. Le réseau est distribué sur deux cartes graphiques, ce qui lui permet d'être plus profond que les réseaux contemporains. Il est constitué de huit couches pondérées : cinq couches convolutives et trois couches entièrement connectées, toutes activées par une fonction ReLU. Les deux premières couches sont suivies de normalisation. Les normalisations et la cinquième couche de convolution sont suivies de *max pooling*. La Figure 3.2 résume l'architecture du réseau.

VGGnet [Simonyan et Zisserman, 2014] obtiennent la seconde place du concours de reconnaissance des images de ImageNet en 2014 avec une TOP-5 précision de 92.7%. Il sert aujourd'hui de base à de nombreux modèles, qui en reprennent l'architecture et souvent même initialisent une partie de leur modèle avec les poids de VGGnet entraîné sur le jeu de données ImageNet afin d'effectuer une tâche de reconnaissance (reconnaître un objet contenu dans une image).

3. http://thoth.inrialpes.fr/workshop/thoth2016/slides/cord.pdf

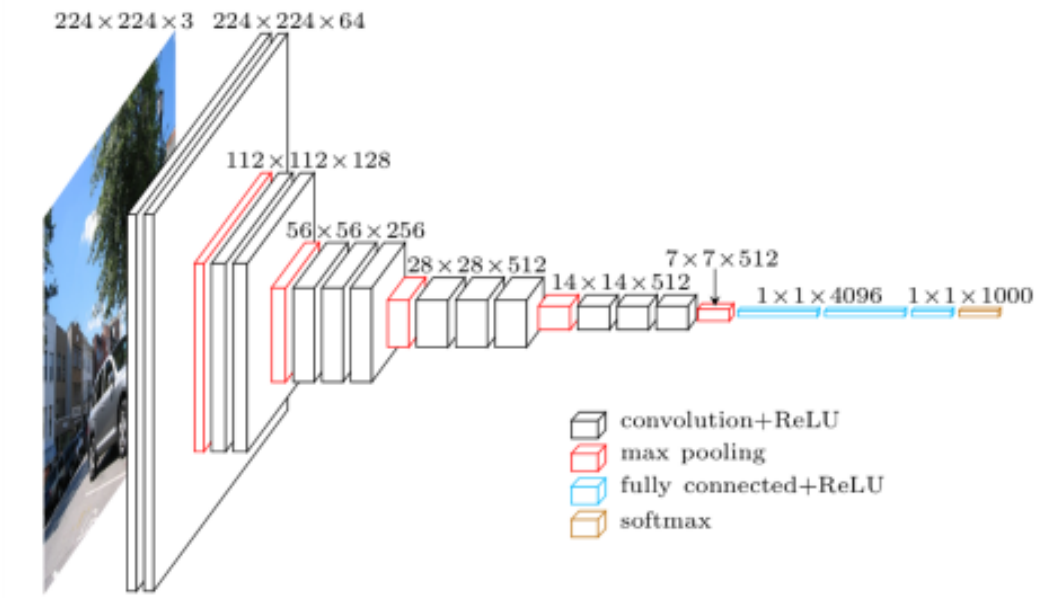

FIGURE 3.3 – Figure extraite de la présentation de Matthieu Cord [3]avec son autorisation. On représente ici l'architecture VGG. Les convolutions permettent d'augmenter la profondeur des *feature maps* et les *poolings* diminuent les dimensions spatiales. En ne comptant pas les couches de *pooling* ou de *softmax* (car elles ne contiennent pas de paramètre à optimiser), le réseau est constitué de 16 couches pondérées (13 de convolutions et 3 complètement connectées). On appelle cette architecture « VGG16 »

Simonyan et Zisserman montrent dans cet article qu'en augmentant la profondeur des architectures utilisées à l'époque on obtenait de meilleures performances. Ce changement leur permet de réduire la taille des fenêtres de convolutions (donc des champs réceptifs) diminuant ainsi le nombre de paramètres à apprendre. Ils proposent ensuite leur propre architecture (VGGnet), illustrée en figure 3.3. Cet architecture peut être adaptée en fonction de la profondeur désirée, donnant lieu à plusieurs variantes dont la plus répandues est VGG16.

ResNet [He *et al.*, 2016] obtiennent la première place du concours de reconnaissance des images de ImageNet en 2016 avec une TOP-5 précision de 96.4%. Ils introduisent le concept de bloc résiduel. Dont l'architecture est illustrée en Figure 3.4.

Le transfert de la sortie de la couche précédente vers la sortie de la couche courante permet deux choses. Le premier avantage est la résolution du problème de fuite du gradient, décrit en Section 2.3.1.2. En effet, s'il est difficile de faire remonter le gradient à travers les couches de convolution, la fonction d'identité permet de contourner ces couches pour aller directement au bloc résiduel précédent. Le deuxième avantage réside dans le fait que chaque bloc doit apprendre le résidu du bloc précédent. En effet, sa sortie $x_n = F(x_{n-1}) + x_{n-1}$ étant la somme de la sortie du bloc précédent x_{n-1} et de la sortie de ses couches convolutives $F(x_{n-1})$, on en déduit que les couches

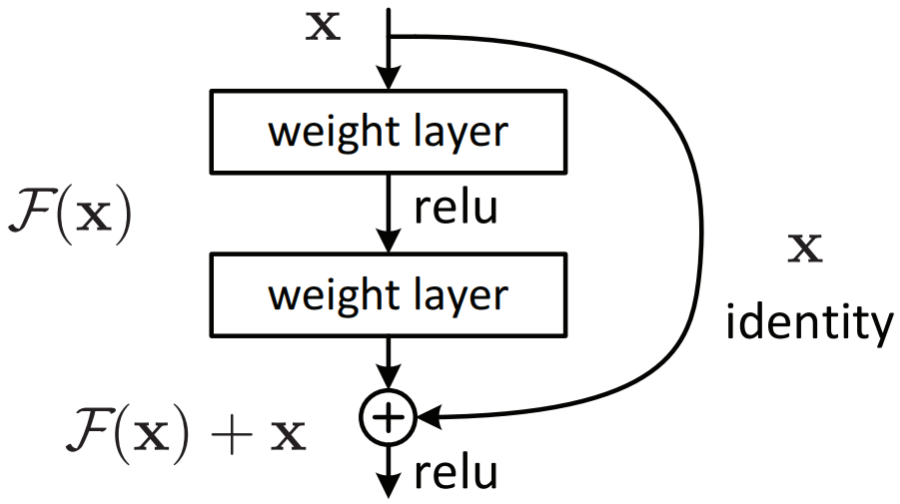

FIGURE 3.4 – Architecture d'un bloc résiduel, figure extraite de l'article original. La sortie du bloc résiduel précédent est utilisée à la fois en entrée de deux couches convolutives, et ajoutée à la sortie de celles ci.

convolutives doivent apprendre à approximer la fonction F comme la différence entre la sortie attendue et la sortie de la couche précédente, $F(x_{n-1}) = x_n - x_{n-1}$. De ce fait, chaque couche ajoutée permet de raffiner la fonction approximée par le modèle et de gagner en performance.

Cette technique rappelle celle du boosting de gradient [Freund, 1995]. C'est une méthode ensembliste utilisant le résidu d'un modèle (plutôt que d'une couche) pour entraîner un nouveau modèle, et ce itérativement autant de fois que nécessaire. Veit *et al.* [2016] montrent que les ResNets se comportent effectivement de manière similaire. Huang *et al.* [2017] utilisent cette connaissance pour entraîner plus efficacement les ResNets en s'appuyant sur les théories de boosting.

3.1.2.3 Encodeur–Décodeur

Les trois architectures présentées servent à la reconnaissance d'image. Elles permettent de savoir ce qui est représenté dans une image, mais ne permettent pas de connaître la localisation des objets reconnus au sein de cette image. Pour obtenir cette information, on applique un algorithme de segmentation sémantique : on ne classifie plus l'image mais chaque pixel qu'elle contient.

Dans l'architecture encodeur–décodeur, la dimension spatiale est réduite petit à petit au fur et à mesure que la profondeur augmente au sein de la partie encodeur, puis la partie décodeur permet de retrouver la dimension spatiale originale. Comme on l'a vu en Section 2.3.2.2, le *pooling* sert à augmenter le champ réceptif, à rendre le modèle moins sensible aux translations, ainsi qu'à réduire la taille des données traitées en divisant, la plupart du temps par deux, la hauteur et la largeur de la carte des propriétés au fur et à mesure que cette dernière gagne en profondeur. C'est la partie

encodeur. Afin de retrouver une segmentation de dimensions similaires à l'image d'entrée, les données passent par une partie décodeur. À l'aide de déconvolutions, ou convolutions transposées, on augmente la largeur et la hauteur de la carte de propriétés. Une déconvolution peut être vue comme une convolution appliquée à une image dont on aurait entouré chaque pixel de pixels nuls, comme illustré en Figure 3.5.

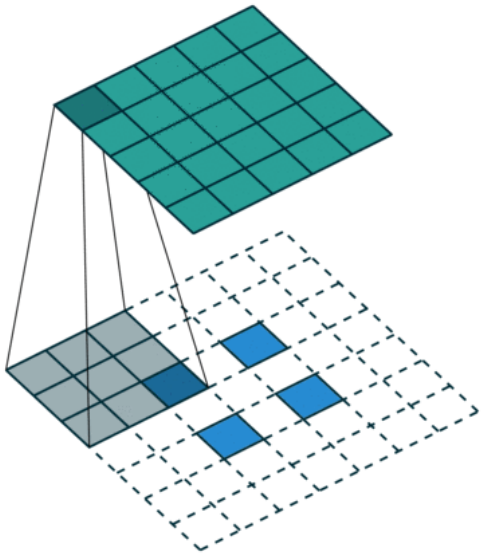

FIGURE 3.5 – Déconvolution. Figure extraite de Dumoulin et Visin [2016]. Le noyau de convolution (en gris) convolue sur la carte de propriétés d'entrée (en bleu) dont les pixels sont entourés de zéros (en blanc). Le résultat est une nouvelle carte de propriétés (en vert) de plus grande résolution.

Fully Convolutional Network (FCN) [Long *et al.*, 2015a] est l'architecture précurseure de la segmentation sémantique. Ses auteurs reprennent les architectures existantes pour la reconnaissance d'image des modèles de référence présentés ci-dessus, pour transformer leurs couches complètement connectées en couches convolutives et ainsi obtenir des modèles de segmentation sémantique (voir figure 3.6) plutôt que des modèles de reconnaissance.

Ils montrent qu'un réseau de neurones permet un apprentissage de bout en bout, c'est à dire qu'ils partent d'une image en entrée du modèle et arrivent à une segmentation en sortie du modèle, sans extraction manuelle de propriété ni transformation de l'image. Ils obtiennent de meilleurs résultats que les algorithmes de l'époque n'étant pas basés sur des réseaux de neurones artificiels.

U-Net [Ronneberger *et al.*, 2015] est un modèle largement utilisé qui propose une telle architecture, décrite en Figure 3.7. Il améliore les déconvolutions utilisées dans le FCN en utilisant les cartes de propriétés de l'encodeur de dimensions équivalentes

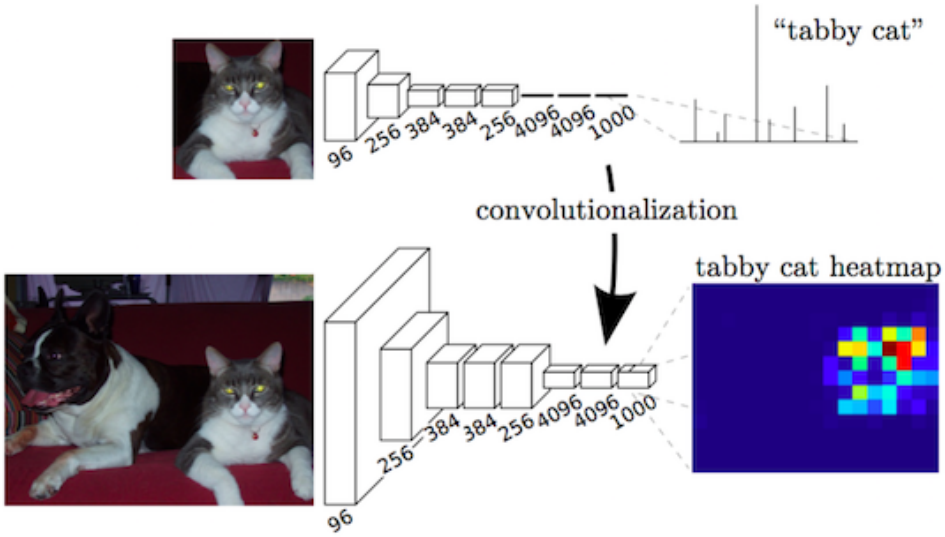

FIGURE 3.6 – On voit ici la transformation d'une architecture type VGG qui se termine par trois couches complètement connectées et produit en sortie une distribution de probabilités permettant de classifier l'image. Dans sa version permettant la segmentation de l'image, les trois dernières couches ont été remplacées par des couches de convolution.

pour parfaire la récupération de la localisation des propriétés. En effet, Lors d'une déconvolution classique, les pixels non nuls sont placés en grille, indépendamment de leur place avant l'opération de *pooling*. De ce fait, les cartes de propriétés résultantes font apparaître des motifs en échiquiers. Les cartes de propriétés pré-*pooling* contiennent moins d'informations sur la nature des objets représentés mais plus d'information sur leur localisation. Le rôle de la déconvolution au sein du U-Net sera d'allier ces connaissances topographiques avec les connaissances extraites de la carte de propriétés de moindre résolution comportant plus d'informations sur la nature des objets représentés.

RefineNet [Lin *et al.*, 2016] est une architecture encodeur décodeur tentant de perfectionner la manière d'utiliser les cartes de propriétés de l'encodeur pour augmenter la résolution des cartes de propriété du décodeur.

Dans cette architecture, l'encodeur est celui de ResNet-101 [He *et al.*, 2016]. Afin d'obtenir des cartes de propriétés d'une résolution de plus en plus grande, le décodeur convolue sur une carte de propriétés de basse résolution sur échantillonnée et sommée avec une carte de propriétés de plus haute résolution. Cette procédure est décrite en figure 3.8 et utilisée à chaque couche du décodeur, comme illustré en figure 3.9.

SegNet [Badrinarayanan *et al.*, 2015] est une architecture ayant choisi une autre solution au même problème. L'indice du pixel retenu par le *pooling* lors de l'encodage est transmis à la couche de déconvolution lors du décodage. La déconvolution

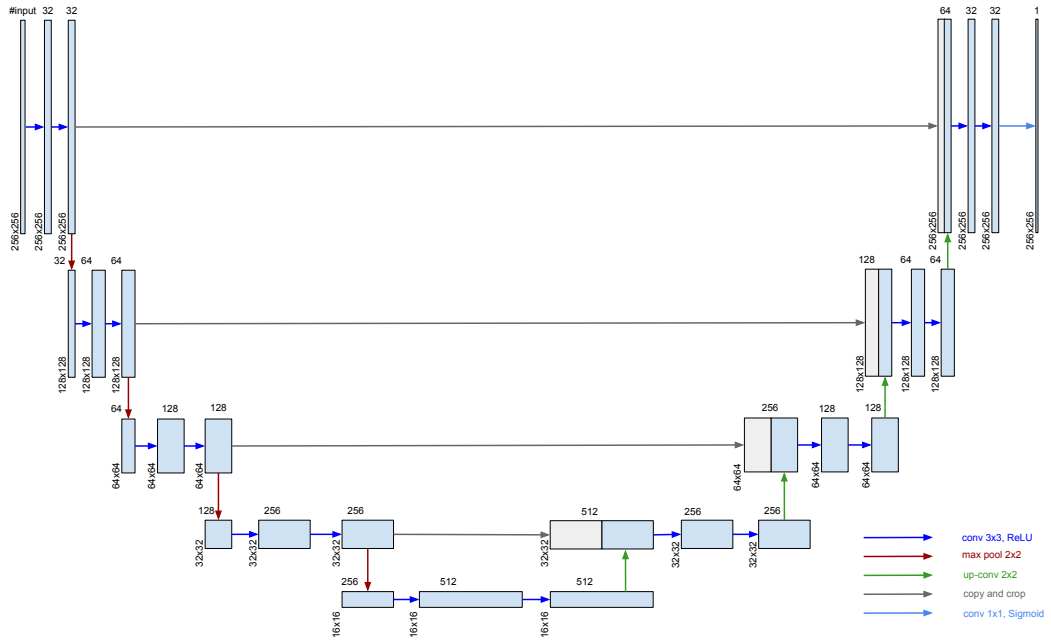

FIGURE 3.7 – Ce schéma reproduit l'architecture U-Net originale. La description de cette figure au sein de l'article original indique que chaque boite bleue correspond à une carte de propriétés multidimensionnelle. Le nombre de canaux est inscrit au sommet de chaque boite, tandis que les dimensions en hauteur et largeur sont indiquées sur l'arrête inférieure gauche de chaque boite. Les flèches dénotent les différentes opérations.

est avantagée par rapport à la version naïve car le pixel non nul est correctement positionné.

Comme le U-Net, il est composé d'un encodeur et d'un décodeur. La différence avec le U-Net est donc que plutôt de transférer toute l'information d'une carte de propriétés de l'encodeur à la carte de propriétés correspondante dans le décodeur, seul un indice par fenêtre de convolution du *pooling* est transféré. C'est un gain en mémoire important pour un faible coût en performance.

3.1.2.4 Convolutions dilatées

Nous venons de voir que les architecture des type encodeur–décodeur tentent de contourner un problème de perte de résolution spatiale due au *pooling*. Yu et Koltun [2015] montrent qu'il n'est pas nécessaire d'avoir recours au *pooling* dans le cadre d'un problème de segmentation. En effet, sacrifier de la connaissance de localisation au profit d'une meilleur reconnaissance est entièrement bénéfique dans le cadre d'un problème de reconnaissance d'image, mais ce n'est pas forcément le cas lorsqu'on souhaite conserver la connaissance de l'emplacement des objets reconnus.

Pour augmenter la taille des champs réceptifs, Yu et al. mettent au point les convolutions dilatées [Yu et Koltun, 2015] (aussi appelées convolutions à trous, ou

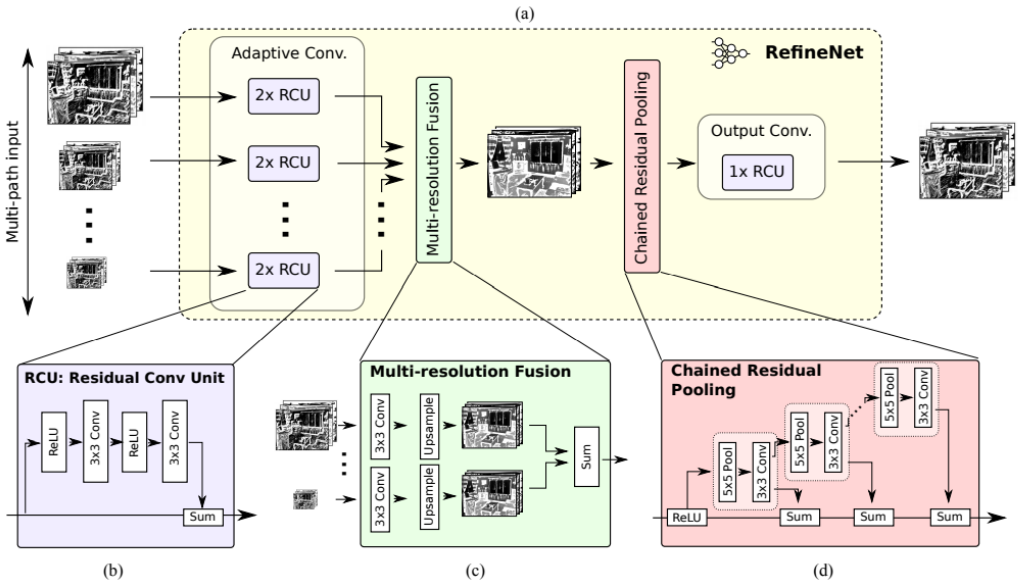

FIGURE 3.8 – Architecture d'une cellule refineNet, extraite de Lin *et al.* [2016]. On voit en (a) l'architecture globale. D'abord chaque entrée (une entrée est une carte de propriétés d'un niveau précédent) traverse deux cellules de convolution résiduelle, afin d'adapter les poids de l'architecture ResNet à la tâche effectuée. Une fusion a ensuite lieu pour ne former qu'une carte de propriétés à partir des entrées. Le bloc final est dit de *poolings* résiduels chaînés. Il est utilisé pour augmenter le champ récepteur de chaque pixel. Une convolution résiduelle finale permet d'obtenir la sortie souhaitée.

atrous convolutions en anglais), utilisant des noyaux de convolution dont les pixels ne sont pas contigus. Ceci permet donc de se passer du *pooling*.

Ils définissent l'opérateur de convolution conventionnel : soit une fonction discrète $F\colon \mathbb{Z}^2 \to \mathbb{R}$. F représentant l'image, qui à une paire de coordonnées, donc un pixel, associe la valeur de ce pixel. Soit $\Omega_r = [-r, r]^2 \cap \mathbb{Z}^2$ et soit un filtre discret $k\colon \Omega_r \to \mathbb{R}$ de taille (en nombre d'éléments) $(2r+1)^2$. L'opérateur de convolution $*$ est défini par

$$(F * k)(\boldsymbol{p}) = \sum_{s+\boldsymbol{t}=\boldsymbol{p}} F(s)k(\boldsymbol{t}) \tag{3.1}$$

où $\boldsymbol{p}$ est le vecteur position contenant les coordonnées dans l'image, s est la position dans l'image associée à t la position dans le filtre, lorsque le filtre est centré sur p. Cet opérateur est appliqué à tous les pixels de l'image. La stratégie de convolution est définie par le sous ensemble auquel appartient p. Si p est pris dans l'ensemble de l'image, la convolution sera appliquée sur tous les pixels. Lorsque ceux-ci sont trop proches du bord, le filtre de convolution k déborde, et il existe donc des p tels que $s + t = p$ et $F(s)$ n'est pas défini. Dans ce cas, il est possible d'agrandir l'image d'entrée (en ajoutant une marge au dimensions souhaitées, soit de pixels nuls, soit

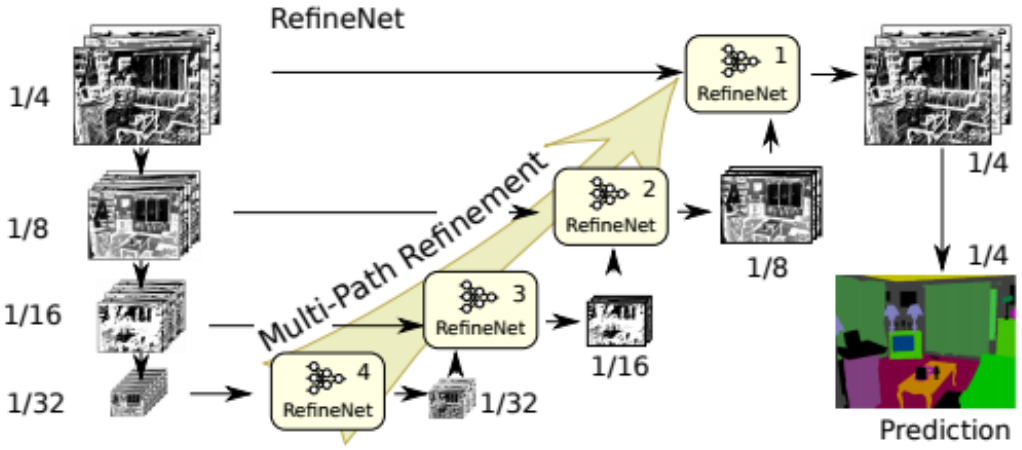

FIGURE 3.9 – Figure extraite de Lin *et al.* [2016].L'encodeur est représenté verticalement sur la gauche du schéma. Le décodeur ramène les cartes de propriétés à la taille originale afin de procéder à une classification de chaque pixel. Pour ce faire, une cellule refineNet permet l'utilisation de cartes de propriétés précédentes, d'une meilleure résolution.

par symétrie, ou autre politique de remplissage au choix), ou il est possible de ne pas calculer le résultat de la convolution pour ces pixels là, amenant à une image plus petite.

Dans le cadre d'une convolution usuelle, on choisira la plupart du temps d'ajouter une marge, que l'on remplira en fonction du problème étudié, afin de conserver la taille de la carte de propriétés, qui sera réduite séparément par des *poolings*. Si on choisit de ne pas ajouter de marge, on peut calculer la taille de l'image résultante. Si l'image d'entrée est de taille $S_{in} = h^2$, et le filtre de convolution de taille $S_f = (2r+1)^2$, alors l'image résultante sera de taille $S_{out} = (h - 2 * r)^2$.

L'opérateur est ensuite généralisé : soit l le facteur de dilatation, l'opérateur de la convolution l-dilatée $*_l$ est défini par

$$(F *_l k)(\boldsymbol{p}) = \sum_{s+l\boldsymbol{t}=\boldsymbol{p}} F(s)k(\boldsymbol{t}) \tag{3.2}$$

avec les mêmes notations. On obtient donc une définition de la convolution plus générale, qui autorise à «dilater» le filtre convolué. Cette convolution est illustrée en Figure 3.10.

Dans le cadre d'une convolution dilatée on peut, si on le veut, réduire la taille de l'image de sortie en n'ajoutant pas de marge à l'image d'entrée. Si l'image d'entrée est de taille $S_{out} = h^2$, et le filtre de convolution de taille $S_f = (2r+1)^2$ avec un facteur de dilatation l, alors l'image résultante sera de taille $S_{in} = (h - 2 * l * r)^2$. À taille de filtre et facteur de dilatation donné, on peut donc déduire la marge nécessaire à l'obtention d'une taille de sortie voulue. Souvent les dimensions de l'image sont conservées.

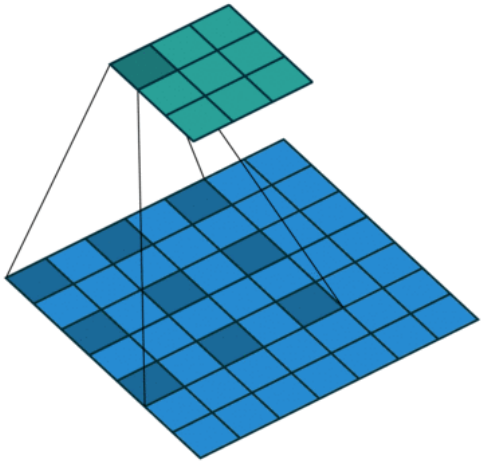

FIGURE 3.10 – Convolution dilatée. Figure extraite de Dumoulin et Visin [2016]. On a ici un filtre de convolution de taille 3 par 3, dilaté avec un facteur 2. Chaque pixel du filtre (bleu foncé) est en effet à distance 2 en largeur et/ou en hauteur du pixel du filtre le plus proche. Le filtre convolue l'image d'entrée (en bleu) pour donner l'image résultante (en vert).

Dans les faits, si les convolutions dilatées sont utilisées pour ne pas perdre d'information spatiale alors la résolution des cartes de propriétés est maintenue. Si Lin *et al.* [2016] reconnaissent que le *pooling* induit une perte d'information spatiale, la convolution dilatée présente selon eux un coût trop important en mémoire. En effet, en augmentant le nombre de carte de propriétés sans réduire les dimensions spatiales, le nombre de paramètres augmente significativement.

3.1.2.5 Agrégation multi-échelles

Comme on l'a vu avec le U-Net et le RefineNet qui agrègent les cartes de propriétés de haute résolution peu informatives de leur décodeur avec les cartes de propriétés de plus basse résolution mais plus informatives de leur décodeur, l'agrégation multi-échelles permet de segmenter finement des objets de tailles différentes. Deux *poolings* sont présentés pour exploiter cette agrégation, le *pooling* pyramidal au sein de l'architecture PSPnet [Zhao *et al.*, 2016] et le *pooling* spatial pyramidal dilaté au sein de l'architecture Deeplab [Chen *et al.*, 2016].

PSPnet est une architecture présentée par Zhao *et al.* [2016] qui se base elle aussi sur celle de ResNet [He *et al.*, 2016]. Des couches de convolutions dilatées [Yu et Koltun, 2015] sont ajoutées. Un module de *pooling* pyramidal est proposé (voir Figure 3.11), consistant en des *poolings* de différentes dimensions dont les sorties sont redimensionnés à la taille de la carte de propriétés d'entrée du *pooling*, puis sont agrégées à cette dernière. Une fonction de coût auxiliaire, en plus de la fonction de coût globale, est appliquée après la quatrième couche de ResNet, juste avant

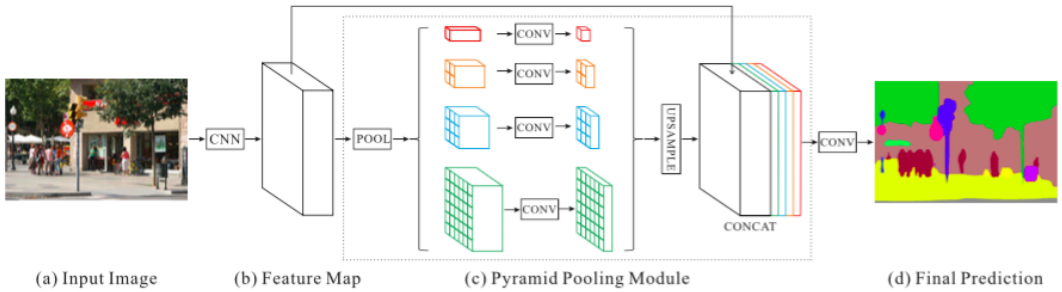

FIGURE 3.11 – Des *poolings* de différentes tailles sont effectuées et des cartes de propriétés de différentes résolutions sont obtenues. Elles sont ensuite sur échantillonnée pour revenir à la résolution pré-*pooling* et concaténées. Cette cellule peut être insérée n'importe où dans le réseau. La différence avec la technique d'*Atrous Spatial Pyramidal Pooling* réside dans le fait que les cartes de propriétés intermédiaires sont obtenues par du *max pooling*, et non pas par des convolutions dilatées.

l'utilisation du *pooling* pyramidal. Elle est utilisée pour s'assurer que la première partie du réseau soit entraînée. Elle sert à optimiser le processus d'entraînement, alors que la fonction de coût globale est responsable de la qualité de la sortie du modèle. Des poids sont alloués aux deux fonctions pour paramétrer leur importance.

Deeplab est une architecture proposée par Chen *et al.* [2016] tirant profit des convolutions dilatées. En s'inspirant du *pooling* spatial pyramida, ils concatènent les cartes de propriétés résultants de convolutions dilatées avec des facteurs de dilatation différents, plutôt que des *poolings* de tailles différentes. Le but est d'avoir des convolutions adaptées à des objets d'échelles différentes. Plusieurs versions de Deeplab existent. Le modèle a été amélioré [Chen *et al.*, 2017b] grâce à l'utilisation de normalisation en batch, qui permet de mitiger le fait que pour de gros facteurs de dilatation les convolutions utilisent beaucoup de pixels situés dans la marge de pixels nuls. Cette marge est ajoutée lorsque l'on veut conserver la résolution de l'image.

3.1.2.6 Récapitulatif

La contribution apportée par chaque modèle est présentée en table 3.1. Ces contributions peuvent aborder le même problème, comme l'agrégation multi-échelle. Dans ce cas là elles sont souvent concurrentes. Lorsqu'elles abordent des problèmes différents, elles peuvent être complémentaires. Ainsi on trouvera des architectures encodeur–décodeur mettant à profit les convolutions dilatées.

Les performances des modèles présentés sont évaluées sur le jeu de données PASCAL Visual Object Classes Challenge [Everingham *et al.*, 2010] qui est un challenge d'analyse d'images ayant eu lieu tous les ans de 2005 à 2012. Il permet de

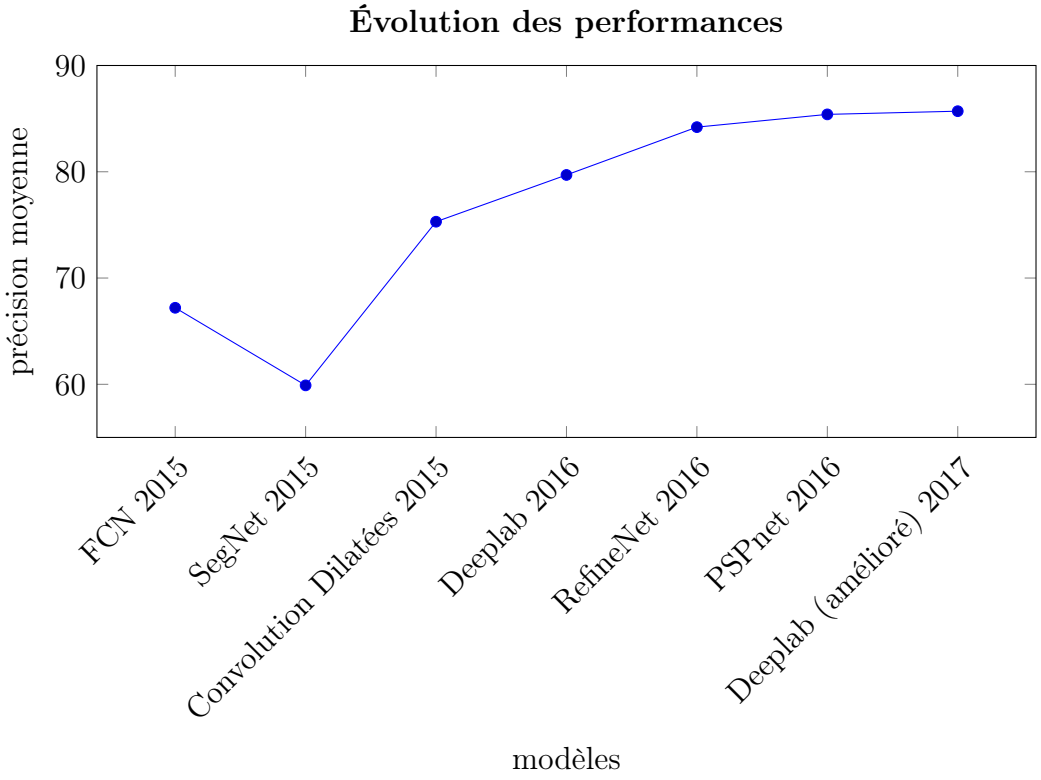

FIGURE 3.12 – On représente ici le score (précision moyenne) obtenu par chaque modèle au challenge VOC de 2012. Les modèles sont classés par ordre chronologique. le score moyen est la moyenne des précisions pour chaque classe.

classer les modèles participant en fonction de leur précision dans les tâches de reconnaissance d'image. Il fournit un jeu d'images, d'annotations (étiquettes) et un procédure d'évaluation qui permettent de standardiser ce classement.

Depuis le décès de Mark Everingham, créateur du projet, la dernière version de 2012 est celle utilisée pour évaluer les algorithmes d'analyse d'image. Dans cette version, 20 classes d'objets différents sont à reconnaître, comme des vélos, des oiseaux, des bouteilles, des chats, des trains...

On représente en figure 3.12 les performances (précision moyenne) obtenues par les modèles présentés ici ayant concouru au challenge de segmentation.

3.1.3 Segmentation d'instances

Nous souhaitons extraire les contours de chaque bâtiment de manière individuelle. La segmentation d'instances permet de répondre à ce problème de manière plus directe que la segmentation. Elle permet non seulement de classifier chaque pixel en fonction de sa classe (arbre, bâtiment, voiture, etc) mais également en fonction de son instance (si c'est un pixel de bâtiment, à quel bâtiment appartient-il ?).

TABLE 3.1 – Contribution apportée par chaque modèle présenté.

Modèle	**Contribution**
AlexNet [Krizhevsky *et al.*, 2012]	Utilisation de carte graphique pour l'apprentissage de réseaux plus profonds
VGGnet [Simonyan et Zisserman, 2014]	Fenêtres de convolutions plus petites pour plus de profondeur
FCN [Long *et al.*, 2015a]	Remplacement des couches complètement connectées par des couches convolutives pour la segmentation
Segnet [Badrinarayanan *et al.*, 2015]	Propagation de l'indice des pixels retenus par le max-*pooling*
U-Net [Ronneberger *et al.*, 2015]	Utilisation des cartes de propriétés de l'encodeur pour reconstruire celles du décodeur
ResNet [He *et al.*, 2016]	Unités résiduelles pour la propagation du gradient
RefineNet [Lin *et al.*, 2016]	Perfectionnement de la fusion des cartes de propriétés
Deeplab [Chen *et al.*, 2016]	Pooling pyramidal spatial dilaté
PSPnet [Zhao *et al.*, 2016]	Pooling pyramidal

Deep watershed [Bai et Urtasun, 2017] est un modèle utilisant une prédiction du masque ainsi que l'image originale pour prédire pour chaque pixel la direction de la bordure de l'instance la plus proche. La distance à cette bordure est prédite dans un second temps. Les auteurs expliquent que la distance aux bordures est un concept trop difficile à apprendre directement. Le principe repose sur l'algorithme de ligne de partage des eaux [Beucher et Meyer, 1992], lequel considère une image à niveaux de gris comme un relief topographique, dont nous simulerions l'inondation. Dès lors que deux étendues d'eaux entrent en contact, nous considérons cette limite comme une bordure entre deux instances. Cette stratégie tend à sur-segmenter l'image, et le modèle proposé par Bai *et al.* vise à apprendre une topographie pour laquelle un

bassin correspond à une instance. Nous avons implémenté leur solution sur des images satellites afin de segmenter les différentes instances de bâtiments. Dans ce jeu de données les bâtiments qui se touchent réellement sont rares, ils sont souvent séparés par au moins un pixel, alors que dans le jeu de données utilisé par les auteurs, le chevauchement des instances est quasi systématique. De ce fait, le modèle n'apprend pas ici à trouver une bordure commune à deux bâtiments mitoyens, mais à trouver le pixel non bâtiment le plus proche. Pour cette raison, nous ne sommes pas parvenus à séparer deux bâtiments trop proches l'un de l'autre en deux bâtiments distincts. Une comparaison entre l'image d'entrée et la vérité terrain produite peut être vue en Figure 3.13 et un exemple de prédiction en Figure 3.14. Dans cette figure, nous choisissons de représenter dans chaque pixel un vecteur en deux dimension : la couche «rouge» contient la composante verticale tandis que la composante horizontale est dupliquée dans les couches «vert» et «bleu» de ces images RGB.

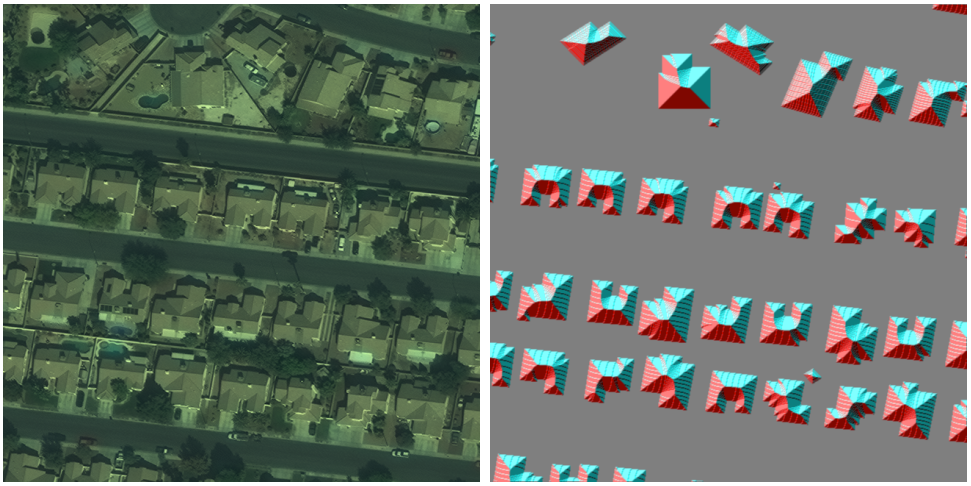

FIGURE 3.13 – Exemple d'image de Vegas et l'étiquette de type direction correspondante

Discriminative loss [De Brabandere *et al.*, 2017] est un modèle dont la fonction de coût oblige le réseau à projeter chaque pixel de l'image dans un espace à n dimensions, de sorte que les vecteurs projetés de pixels appartenant à la même instance soient proches les uns des autres, tandis que les vecteurs projetés de pixels appartenant à des instances différentes sont éloignés les uns des autres. La sortie du réseau peut facilement être partitionnée avec une opération de post-traitement simple et rapide, basée sur un seuillage de la distance entre éléments d'une même partition. Les auteurs fournissent une implémentation [4] de leur modèle sur pytorch [Paszke *et al.*, 2017]. Tout notre processus d'évaluation reposant sur l'utilisation de Keras [Chollet *et al.*, 2015] afin de profiter de tout l'environnement déjà développé par nos soins pour entraîner nos autres modèles, nous n'avons pas pour l'instant évalué les performances de ce modèle. Dernièrement, une implémentation compatible avec Keras a

4. https://github.com/DavyNeven/fastSceneUnderstanding

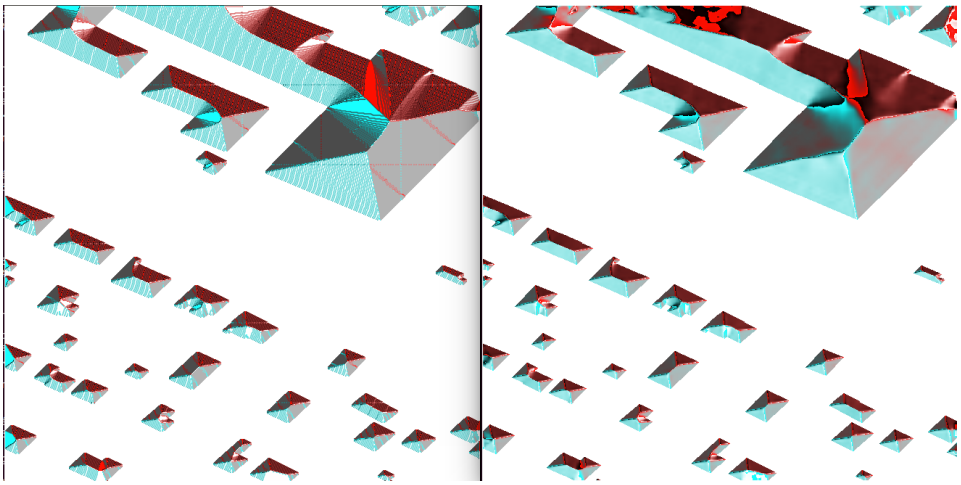

FIGURE 3.14 – Exemple de prédiction du vecteur de direction indiquant la bordure la plus proche. L'étiquette est à gauche, la prédiction à droite.

été publiée sur Github [5], ce qui nous permettra une évaluation prochaine.

Mask R-CNN [He *et al.*, 2017] est un modèle de segmentation d'instance reposant sur un principe simple : la segmentation sémantique de la boîte englobante d'un objet (en deux classes : objet englobé ou non) résulte en une segmentation d'instance. Le modèle proposé est composé de plusieurs modules. Le premier module est celui de Fast R-CNN [Girshick, 2015], permettant de proposer des régions d'intérêt. Un deuxième module, un petit FCN, est chargé de segmenter chaque région d'intérêt, en parallèle. Finalement, une branche de Fast R-CNN prédit la classe de l'objet segmenté. L'apprentissage de ce modèle requiert un à deux jours d'entraînement sur une machine à 8 GPUs. Ne disposant pas d'une telle machine, nous n'avons pas évalué Mask R-CNN sur notre jeu de données pour l'instant. Zhao *et al.* [2018] ont utilisé ce modèle pour extraire des contours de bâtiments d'images satellite, obtenant une amélioration significative par rapport à l'état de l'art. Ils proposent également un post-traitement menant à de moins bonnes performances, mais à des polygones plus réguliers.

3.1.4 État de l'art de la détection des contours de bâtiments dans les images satellites

L'automatisation de la production de cartes basée sur l'analyse d'images aériennes a d'abord été étudiée par Hunt *et al.* [1991] quand ils ont utilisé des réseaux neuronaux pour faciliter un processus manuel de tracé. Le but est d'utiliser les données brutes à notre disposition, principalement des images satellites, pour obtenir la carte d'une région donnée. Les auteurs ci-dessus l'ont amélioré par l'utilisation d'un

5. https://github.com/hq-jiang/instance-segmentation-with-discriminative-loss-tensorflow

autre réseau neuronal [DeKruger et Hunt, 1994] pour classer les motifs de texture. Les deux réseaux sont alimentés par des caractéristiques extraites d'images satellites, les CNN n'étant pas encore d'usage.

Ahmadi *et al.* [2010] proposent un modèle de contour actif pour détecter les bâtiments. Ils testent leur méthode sur une seule image. Ce modèle implique des courbes ou des surfaces dynamiques qui se déplacent à l'intérieur d'un domaine d'image pour capturer les caractéristiques désirées de l'image. Cette technique ne détecte initialement pas que les bâtiments, mais tout objet dont la surface est de couleur ou de texture homogène. Pour le limiter à la détection de bâtiments, ils ajoutent à leur fonction d'erreur un terme permettant de distinguer les bâtiments des autres objets. Ils comparent les pixels des objets détectés à des pixels «témoins», sélectionnés à la main à l'intérieur des bâtiments.

Vakalopoulou *et al.* [2015], et Chen *et al.* [2017a] ont tous deux conçu des modèles essayant d'extraire les contours des bâtiments des images satellites. Dans le premier cas, ils ont utilisé un CNN pour classer les patchs comme contenant des bâtiments ou non, puis ils ont effectué une segmentation en patchs avec un SVM [Cortes et Vapnik, 1995]. Ils extraient les contours après la segmentation et évaluent leur modèle sur des images Quickbird et Worldview 2, pour la ville d'Attica, en Grèce. Ils ne précisent pas ce qu'est un vrai positif, *i.e.* comment considérer qu'un contour extrait correspond au contour extrait manuellement (vérité terrain). Ce manque de clarification empêche de comparer leur modèle à d'autres. Leur modèle est évalué sur trois images, contenant un total de 1438 bâtiments seulement.

Chen *et al* ne donnent pas le nom de leur ensemble de données, ce qui entrave également la comparaison. Ils utilisent un CNN pour ajuster la taille et l'orientation des rectangles au-dessus des bâtiments. Ils ne donnent la performance de leur modèle que sur 4 images choisies à la main parmi leurs 2000 images disponibles.

Yuan [2017] propose un CNN architecturé pour prédire pour chaque pixel de l'image la distance au contour le plus proche du bâtiment. En affichant la plus petite des distances, ils peuvent afficher les contours. Cependant, l'objet polygone associé à cette représentation n'est jamais créé, donc la précision et le rappel sont calculés en fonction de la classification des pixels, pas de la détection des bâtiments.

Dans les études précédentes, la segmentation des pixels a souvent été l'étape finale envisagée et seules quelques propositions atteignent l'objectif final qui consiste à extraire des objets. Dans le premier cas, chaque pixel de l'image se voit attribuer une classe, correspondant à l'objet représenté dans le pixel. Il donne des informations sur le type d'objet représenté par l'image et l'emplacement de ces représentations. Cependant, contrairement à l'extraction de contours, il n'y a aucune notion de l'instance d'un objet. En passant par la phase de détection d'objets, on se retrouve avec des instances d'objets ; dans notre cas, les bâtiments sont représentés par des polygones. Cette dernière étape est obligatoire pour produire des cartes ou corriger des cartes existantes. Avec seulement des pixels, on ne peut pas compter les bâtiments, connaître le contour d'un bâtiment, ou attribuer des propriétés à des bâtiments tels qu'une adresse ou une fonction (magasin, station-service...). Une étape finale consistant à créer les bâtiments en tant qu'entités à partir de la classification des pixels est nécessaire.

Pour comparer ces modèles, on utilise la métrique F1-score, décrite en Sec-

tion 3.2.2. Le tableau 3.2 résume les modèles existants pour la détection des contours de bâtiments. On notera que les F1-Scores ne sont pas utilisables pour une comparaison rigoureuse des modèles. En effet, ils ont été calculés sur très peu d'images, parfois une seule, ou sur une sélection des meilleures images de l'ensemble de données utilisé.

TABLE 3.2 – Résumé des techniques connues de détection des contours de bâtiments

Référence	**Solution**	**Jeu de données**	**F1-Score**
Ahmadi et al. Ahmadi *et al.* [2010]	Contours actifs	Lavasan	0.847
V. et al.Vakalopoulou *et al.* [2015]	patch classification via CNN + segmentation SVN	Attica	0.847
Chen et al.Chen *et al.* [2017a]	rectangles ajustés par CNN	Non nommé	0.976
Yuan et al.Yuan [2017]	distance par CNN	Washington	Inconnu

3.2 Challenges SpaceNet

Les challenges SpaceNet [6] sont des concours en ligne, sponsorisés par DigitalGlobe, CosmiQ Works, et NVIDIA. Ils proposent aux participants de mettre au point des modèles dont le but est d'extraire des contours de bâtiments depuis des images satellites. Une version ultérieure propose également l'extraction du réseau routier [7]. La participation à ces challenges alors que nous débutions nos travaux sur l'analyse d'images satellites pour la correction d'OSM fut une aubaine. Nous avons ainsi disposé à la fois d'un jeu de données étiqueté, que nous présentons en Section 3.2.1, et de l'opportunité de mesurer nos progrès face à des concurrents motivés,

6. https://spacenetchallenge.github.io/
7. https://spacenetchallenge.github.io/Competitions/Competition3.html

grâce à un protocole décrit en Section 3.2.2. Cette section présente en détails ces concours qui furent le fil rouge de nos travaux.

3.2.1 Données

Cette section présente les données fournies dans le contexte des deux SpaceNet Challenge relatifs à la détection des bâtiments. Nous décrivons d'abord les images satellites puis les étiquettes associées.

3.2.1.1 Images

Deux éditions du challenge ont eu lieu. La première portait sur le jeu de données de Rio, au Brésil. Deux type d'images satellites sont mises à disposition pour ce challenge, des images 3 bandes RGB et des images 8 bandes. Les données d'apprentissage contiennent 7000 images, chaque image couvre 200m x 200m au sol. Les images 3 bandes ont une résolution d'environ 50 cm, les images 8 bandes ont une résolution d'environ 2 m. Les images 8 bandes couvrent les longueurs d'onde $\{427, 478, 546, 608, 659, 724, 833, 949\}$ nanomètres, ce qui permet de différencier les matériaux. Les images sont fournies au format GeoTiff.

La deuxième édition portait sur les villes de Vegas, Paris, Shanghai, et Khartoum. Ce challenge propose des étiquettes de meilleure qualité et différents types d'images. Les images RGB-PanSharpen sont l'équivalent des images RGB de la première version. Elles ont cependant une résolution de 30cm par pixel plutôt que 50. Les images MUL, équivalente des 8 bandes précédentes ont une résolution de 1.2m plutôt que 2m. Viennent s'ajouter les images PAN, une seule couche de résolution 30cm et MUL-PanSharpen, 8 canaux d'une résolution de 30cm. Plus de 10 000 images sont disponibles cette fois. Dans cette édition, des données extérieures pouvaient être utilisées, notamment les données d'OSM. Ces données sont géographiques, des lignes et des polygones semblables aux polygones des bâtiments. Il faut donc les transformer en image pour pouvoir les utiliser. Elles sont transformées en masque (des cours d'eaux, des routes, des bâtiments, etc) et concaténées aux autres couches.

Le workshop deepglobe de la conférence CVPR 2018 reprend les données et les modalités de ce second challenge SpaceNet. Il fut le cadre de nos expérimentations sur la fusion de segmentations qui fait l'objet de la Section 3.4.

Les dimensions des jeux de données pour chaque ville sont données dans la Table 3.3.

3.2.1.2 Étiquettes

La vérité terrain associée, le contour des bâtiments, est fournie dans un fichier CSV contenant l'identifiant de l'image, l'identifiant du bâtiment, les coordonnées du polygone au format Well-Known-Text [8] (WKT) utilisant les coordonnées en pixel au sein de l'image, et les coordonnées du polygone au format WKT utilisant les coordonnées géographiques. Lorsqu'un bâtiment contient un trou, il ne peut pas être représenté sous la forme d'un simple polygone. Dans ce cas, il est composé de plusieurs

8. http://www.opengeospatial.org/standards/wkt-crs

TABLE 3.3 – Dimensions des jeux de données pour chaque ville

Ville	**# Images**	**# Bâtiments**	**Surface (km^2)**
Rio	7 000	382 534	2 544
Vegas	3 851	151 367	216
Paris	1 148	23 816	1 030
Shanghai	4 575	92 015	1 000
Khartoum	1 012	35 503	765

polygones, le premier représentant le contour extérieur et les suivants le contour des trous, ce qui est contraire au standard WKT. Un extrait de vérité terrain est présenté en Table 3.4.

TABLE 3.4 – Exemple de polygones contenus dans la vérité terrain, extraits de l'énoncé du challenge.

ImageId	**BuildingId**	**WKT_Pix**	**WKT_Geo**
AOI_1_RIO_img1	1	"POLYGON ((103 205 0,[...],103 205 0))"	"POLYGON ((-43.6816 -22.9812 0, [...]))"
AOI_1_RIO_img2	-1	POLYGON EMPTY	POLYGON EMPTY
AOI_1_RIO_img3	1	"POLYGON ((213 269 0,[...],151 253 0))"	"POLYGON (([...]))"

Il est précisé que certains polygones ne sont pas alignés avec leur représentation dans l'image satellite, que certains bâtiments ont été oubliés, et qu'il existe des polygones dans la vérité terrain ne correspondant à aucun bâtiment. Aucune quantification de ces erreurs n'est fournie.

3.2.2 Évaluation

L'évaluation se fait de la même manière pour les deux itérations du challenge. Un jeu d'images non étiquetées est fourni, de taille et de contenu équivalent au jeu de données étiqueté. Les participants doivent produire les polygones au format WKT utilisant les coordonnées pixel et les assembler au sein d'un fichier CSV contenant l'identifiant de l'image, l'identifiant du bâtiment, le polygone, un indice de confiance pour le polygone.

Pour évaluer la correspondance entre deux polygones P_a et P_b, le score *Intersection Over Union* (IoU) est calculé :

$$IoU(P_a, P_b) = \frac{area(P_a \cap P_b)}{area(P_a \cup P_b)} \tag{3.3}$$

L'algorithme d'évaluation est alors le suivant :

```
Trier polygones_predits par indice de confiance décroissant.
vrais_positifs = 0
faux_positifs = 0
faux_negatifs = 0
pour chaque polygone_predit dans polygones_predits:
    max_IoU = 0
    meilleur_polygone_etiquette = None
    pour chaque polygone_etiquette dans polygones_etiquettes:
        si polygone_predit.ImageId == polygone_etiquette.ImageId et
                    !polygone_etiquette.reserve:
            current_IoU = IoU(polygone_predit, polygone_etiquette)
            si current_IoU > max_IoU :
                max_IoU = current_ioU
                meilleur_polygone_etiquette = polygone_etiquette
            finsi
        finsi
    finpour
    si max_IoU >= 0.5:
        polygone_predit.correspondant = meilleur_polygone_etiquette
        meilleur_polygone_etiquette.reserve = Vrai
        vrais_positifs += 1
    sinon:
        faux_positifs += 1
    finsi
finpour
pour chaque polygone_etiquette dans polygones_etiquettes:
    si !polygone_etiquette.reserve:
        faux_negatifs += 1
    finsi
finpour
precision = vrais_positifs / (vrais_positifs + faux_positifs)
rappel = vrais_positifs / (vrais_positifs + faux_negatifs)
retourner 2 * precision * rappel / (precision + rappel)
```

Le score retourné par cet algorithme s'appelle le F1-score [Sasaki *et al.*, 2007]. Nous utiliserons ce score pour l'évaluation de tous les modèles au sein de ce chapitre.

3.3 Étude comparative des représentations de la vérité terrain

Cette section décrit nos expérimentations relatives à la transformation des polygones d'un format textuel à une image. Nous y mettons en évidence la multiplicité des transformations possibles. Nous comparons trois transformations des étiquettes en image. Nous mettons en évidence que pour une architecture donnée, les différents types d'étiquettes ne permettent pas d'atteindre des performances équivalentes.

Le traitement des données se fait en trois étapes. Il faut d'abord transformer la vérité terrain de manière à pouvoir l'utiliser pour le processus d'apprentissage. Pour une segmentation, on détermine le type d'images à produire à partir des polygones fournis, que notre modèle tentera de reproduire en ne se basant que sur les images satellites. Il faut ensuite sélectionner un modèle et l'entraîner avec les images satellites et la vérité terrain produite précédemment. Une fois les images segmentées, reste à extraire les contours des bâtiments de la segmentation.

3.3.1 Transformation des étiquettes

Les étiquettes fournies se présentent sous forme d'un fichier CSV contenant des polygones dans un format textuel tandis que les étiquettes utilisées pour la segmentation sont habituellement sous forme d'image. Plusieurs types d'images peuvent être produits à partir d'un ensemble de polygones. Il est possible de représenter les polygones directement sous forme de traits. Cette représentation est potentiellement ambiguë, car elle ne distingue pas le contour extérieur d'un bâtiment et un polygone intérieur représentant un trou. De plus, rien n'oblige le modèle à produire des polygones, les contours produits ne seront pas forcément fermés. Trois transformations alternatives ont été implémentées : le masque, le champ de distance, et ce qu'on appellera ici les étiquettes à 3 classes. Un exemple de chacune de ces transformations est illustré en Figure 3.15. Pour chacune de ces transformations, la même architecture de modèle peut être utilisée. Seul le nombre de sorties de la dernière couche est modifié.

Le masque indique si chaque pixel représente une partie de bâtiment (valeur de 1) ou de fond (valeur de 0). C'est le type d'étiquette utilisé par défaut dans la plupart des modèles de segmentation. Un modèle entraîné à produire des masques va prédire pour chaque pixel la probabilité d'être un pixel de bâtiment. Cette représentation est facile à produire, à apprendre et à évaluer. Le principal défaut de cette représentation est l'impossibilité de distinguer deux bâtiments mitoyens d'un seul bâtiment. On voit en Figure 3.15 que contrairement aux deux autres représentation (à droite de l'image), la représentation sous forme de masque (en bas à gauche) fusionne la plupart des bâtiments en un seul.

Le champ de distance est la transformation utilisée par Yuan [2016] qui associe à chaque pixel une valeur représentant sa distance, en pixels, à la bordure de bâtiment la plus proche. Cette distance est négative à l'extérieur des bâtiments et positive

à l'intérieur. Cette représentation très riche permet de séparer les uns des autres, de distinguer les bordures (valeurs proches de 0), les bâtiments (valeurs positives) et le fond (valeurs négatives). Cependant, nous verrons qu'il peut être plus difficile d'entraîner un modèle à prédire ces distances. Yuan regroupe les valeurs des distances en 128 classes, représentant les valeurs de -64 à 63.

Les étiquettes à 3 classes permettent de distinguer la classe «bâtiment», la classe «fond» et la classe «bordure». Ces étiquettes ont été spontanément utilisées par les gagnants du premier challenge SpaceNet. Elles sont produites à partir du champ de distance. L'épaisseur de la bordure est fixée en choisissant la plage de valeurs de distance appropriée. Les bordures ne peuvent pas être définies par les valeurs négatives (dans le fond), sinon deux bâtiments mitoyens n'auraient pas de bordure. Elles ont donc des valeurs positives (vers l'intérieur du bâtiment). Si elles sont trop petites, le modèle ne parviendra pas à apprendre. En effet, l'imprécision des étiquettes conduit à un faible chevauchement entre les bordures réelles et les bordures données, à quelques pixels prêt. Si elles sont trop grosses alors la plupart des petits bâtiments ne sont plus classifiés que comme bordure, plus du tout comme bâtiments. Une bordure de 4 pixels a été choisie empiriquement, permettant d'entraîner le modèle tout en gardant des pixels de classes bâtiment à l'intérieur des petits bâtiments. Chaque pixel a pour valeur 1 dans la classe associée et 0 dans les deux autres. Marmanis *et al.* [2018] ont montré postérieurement à notre étude que la création d'une classe de bordure a significativement amelioré leur segmentation d'images satellites.

Les directions des bordures ont été introduites par Bai et Urtasun [2017] postérieurement à notre étude. Comme nous l'avons vu en Section 3.1.3, nous n'avons pas pu les utiliser à cause de spécificités du jeu de données. Ces étiquettes ont été étudiées pour des jeux de données dans les quels les instances se chevauchent presque toujours, quand notre jeu de données présente des instances de bâtiments étant séparées les unes des autres dans la majorité des cas.

3.3.2 Protocole expérimental

Cette section présente le protocole d'évaluation des modèles puis décrit les modèles auxquels nous nous comparons.

3.3.2.1 Évaluation des modèles

Les données utilisées pour entraîner et évaluer les modèles sont les données du premier challenge SpaceNet. Nous utilisons donc des images RGB d'une résolution de 50cm par pixel, dans lesquelles figurent les bâtiments de la ville de Rio. Sur les 7 000 images fournies, 60% sont utilisées dans l'ensemble d'apprentissage, 20% dans l'ensemble de validation et 20% dans l'ensemble de test. Chaque modèle est entraîné pendant 100 époques. Celle obtenant la meilleure précision sur l'ensemble de validation est sélectionnée pour produire les prédictions sur le l'ensemble de test.

L'architecture utilisée pour la segmentation est l'architecture SegNet [Badrinarayanan *et al.*, 2015]. Ce choix fut motivé par le fait qu'en 2016, elle était commune et

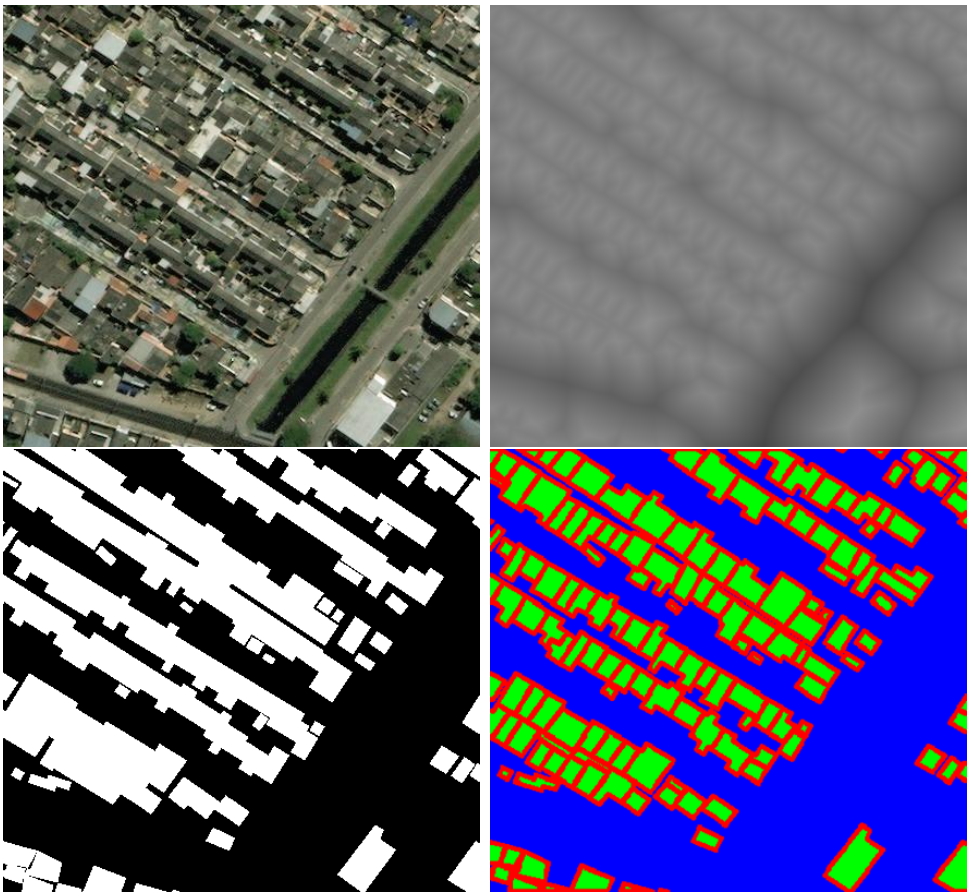

FIGURE 3.15 – Transformation des étiquettes. La première image est l'image satellite RGB correspondant aux étiquettes. En haut à droite le champ de distance extrait des polygones étiquettes. En bas à gauche le masque correspondant. En bas à droite une étiquette de segmentation à trois classes, intérieur de bâtiment, fond, et bordure, avec une bordure de taille 4 pixels.

son implémentation disponible pour toutes les bibliothèques d'apprentissage profond. Avec cette architecture, nous avons évalué et comparé la prédiction de trois types d'images ; masques, distance aux bordures, trois classes. Le modèle est implémenté grâce à la bibliothèque Keras [Chollet *et al.*, 2015], et la fonction d'erreur utilisée est l'entropie croisée. Si le nombre de classes est de deux, la formule de l'entropie croisée est la suivante :

$$\begin{aligned} \hat{y} &= f(\boldsymbol{X}; \boldsymbol{w}) \\ J(\boldsymbol{w}; f) &= -(y * log(\hat{y}) + (1 - y) log(1 - \hat{y})) \end{aligned} \tag{3.4}$$

où f est le modèle dont la prédiction $\hat{y}$ sur son entrée $\boldsymbol{X}$ (la probabilité que l'entrée appartienne à la classe donnée), paramétrée par $\boldsymbol{w}$, est comparée par la fonction de coût J à l'étiquette y. Avec les mêmes notations, Lorsque la classification se fait entre C classes avec $C > 2$, la formule de l'entropie croisée devient :

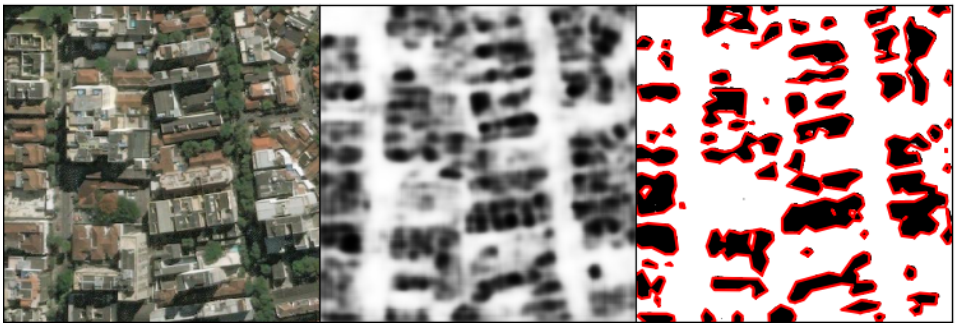

FIGURE 3.16 – Extraction des contours de bâtiments depuis la prédiction du masque

$$-\sum_{c=1}^{C} y_c * log(\hat{y}_c) \tag{3.5}$$

Une fois la segmentation effectuée, et pour chacune des trois transformations possibles, on extrait les contours en transformant l'image prédite en masque :

— Pour les masques prédits aucun traitement n'est nécessaire.
— Pour la distance, on extrait les pixels avec une distance prédite positive.
— Pour les étiquettes à trois classes, on extrait la classe «bâtiment».

Ensuite, un seuil est appliqué à 0.5, valeur au delà de laquelle on considère le pixel comme un pixel de bâtiment et en dessous de laquelle on considère le pixel comme un pixel de fond. Les polygones sont créés grâce à la fonction *find_contours* de la bibliothèque scikit-image [van der Walt *et al.*, 2014], une adaptation en 2D de l'algorithme *Marching Cubes* [Lorensen et Cline, 1987]. Un exemple est illustré en Figure 3.16.

On appelle par la suite SegMask l'architecture SegNet responsable de la prédiction des masques, SegDist l'architecture SegNet responsable de la prédiction des distances, et SegBorder l'architecture SegNet responsable de la prédiction des trois classes bâtiment, bordure, fond. Ces architectures ont été développées pendant la durée du premier challenge SpaceNet, comme le furent les deux premiers modèles concurrents.

3.3.2.2 Modèles concurrents

On compare ces trois modèles aux modèles suivants : Wleite et Cygan qui sont les champion et vice champion du premier SpaceNet challenge. Leur code et la description de leur solution sont disponibles sur Github [9], et Yuan [Yuan, 2016] qui représente l'état de l'art et introduit la prédiction de la distance.

Wleite a proposé la solution ayant remporté le challenge. Elle est basée sur des forêts aléatoires [Svetnik *et al.*, 2003] segmentant l'image en 3 classes. Contrairement

9. https://github.com/SpaceNetChallenge/BuildingDetectors

à l'apprentissage profond, les forêts aléatoires requièrent une extraction manuelle des propriétés de l'image. Les arêtes verticales et horizontales sont extraites de chacune des couches des images 8 bandes et d'une image en niveau de gris fabriquée depuis l'image RGB en convoluant les matrices $\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$ et $\begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}$.

Les pixels RGB sont convertis en TSL, pour Teinte, Saturation, lumière, un autre espace de représentation des couleurs. En tout, 96 propriétés sont extraites de zones de pixels de tailles variables (de fenêtres carrées de taille 2 à 16). Elles sont décrites par la moyenne, la variance et l'asymétrie des valeurs des pixels contenus, et la valeur de la détection d'arête. Un pixel sur 5 est utilisé. Une forêt aléatoire classifie chaque pixel comme bâtiment ou non, une autre comme bordure de bâtiment ou non. En retirant les pixels de bordure, Wleite utilise un algorithme de remplissage par diffusion paramétré par un seuil, pour séparer les bâtiments les uns des autres. Les polygones décrivant l'enveloppe convexe de chaque groupe de pixels connexe sont utilisés comme contour des bâtiments.

Cygan a proposé la meilleure solution basée sur de l'apprentissage profond. Elle a atteint la deuxième place du classement. Elle repose sur deux réseaux de neurones. Le premier, responsable de la segmentation en 3 classes des images est une architecture à base de convolutions dilatées. Le post-traitement est un réseau de neurones dont le but est de prédire si un des 16 rectangles prédéfinis correspond à un bâtiment dans une zone de l'image. Cette dernière est au préalable découpée en cases de 50x50 pixels. Si un rectangle est attribué à un bâtiment, il est ajusté par des régressions qui prédisent la largeur, hauteur et translation optimale, ainsi qu'un angle de rotation à appliquer. Ce post-traitement est inspiré de Liu *et al.* [2016].

Yuan propose une architecture basée sur trois convolutions successives, chacune suivie de max-pooling. Les sorties de chacune de ces convolutions sont sur-échantillonnées via une interpolation bi-linéaire pour retrouver la taille de l'entrée. elles sont ensuite concaténées. Une dernière convolution est appliquée à cette carte de propriétés pour obtenir la segmentation finale. Cette architecture est représentée en Figure 3.17. Le code de Yuan n'étant pas public, nous avons développé cette architecture conformément à la description fournie dans la publication [Yuan, 2016].

3.3.3 Résultats

10. https://github.com/SpaceNetChallenge/BuildingDetectors/tree/master/wleite
11. https://github.com/SpaceNetChallenge/BuildingDetectors/tree/master/marek.cygan

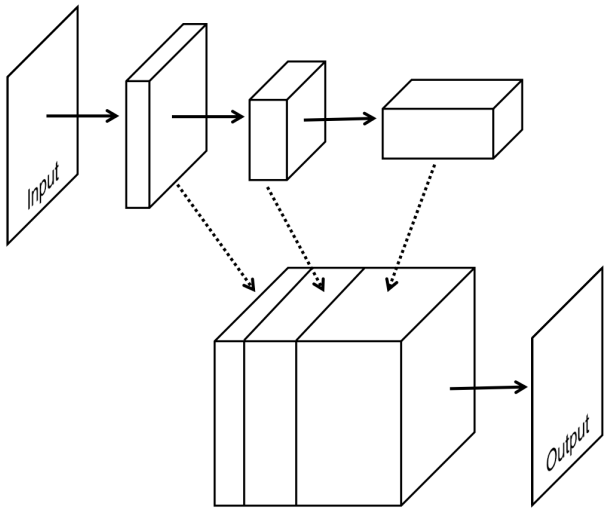

FIGURE 3.17 – Figure extraite de Yuan [2016] décrivant l'architecture du modèle proposé par Yuan.

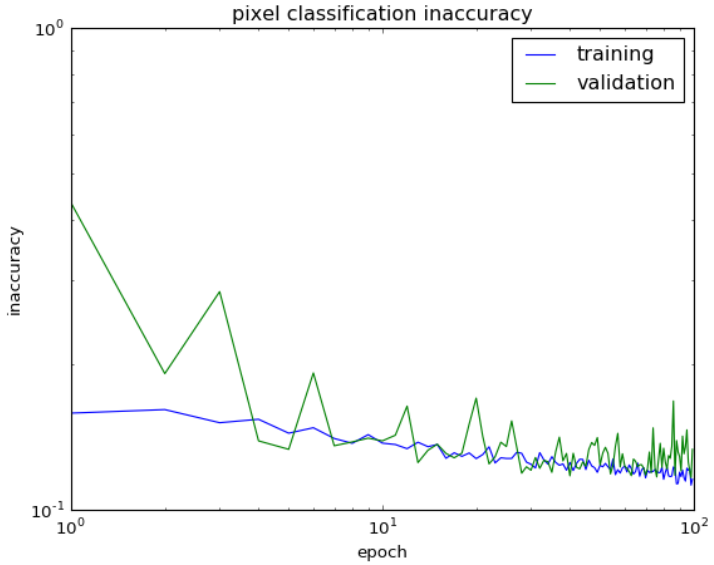

FIGURE 3.18 – Évolution de l'imprécision (1-précision) lors de l'apprentissage du modèle SegNet sur des étiquettes à 3 classes. On voit que le modèle cesse de s'améliorer sur les données de validation aux alentours de l'époque 40.

Afin d'illustrer le processus de sélection d'époque, un exemple d'évolution des performances au cours de l'apprentissage, ici sur le problème à trois classes, est tracé en Figure 3.18. On voit qu'autour de l'époque 40 le modèle commence à sur-apprendre.

Le tableau 3.5 présente les résultats de différents modèles. En comparant avec les résultats des autres concurrents, l'architecture SegNet s'est révélée peu satisfai-

TABLE 3.5 – Récapitulatif des performances de différents modèle sur le jeu de données du premier challenge SpaceNet.

Modèle	**Étiquettes**	**Architecture**	**Post-traitement**	**F1-Score**
Wleite [10]	3 classes	Forêt aléatoire	remplissage	0,255
Cygan [11]	3 classes	Dilated FCN		0,245
Yuan [Yuan, 2016]	Distance	FCN	Seuil	0,168
SegBorder	3 classes	SegNet	Seuil	0,168
SegMask	Masque	SegNet	Seuil	0,137
SegDist	Distance	SegNet	Seuil	0

sante : la prédiction de la distance a par exemple échoué alors qu'elle était possible avec l'architecture proposée par Yuan [2016]. On a vu en Section 3.1.2.6 que l'architecture SegNet n'est pas la plus performante. On voit dans la Figure 3.19 que le sur échantillonage n'a pas permis de retrouver une résolution permettant d'extraire des bâtiments. Une erreur d'implémentation ne pourrait être exclue qu'en retrouvant les mêmes performances que celle donnée dans l'article introduisant cette architecture, sur le jeu de données ImageNet. Nous n'avons pas eu l'occasion de recourir à de telles vérifications. Mise à part cette possibilité, on peut supposer que l'agrégation multi-échelle des cartes de propriétés aux dimensions variées permet un meilleur sur-échantillonage.

La prédiction des étiquettes à trois classes s'est montrée significativement plus performante que la prédiction du masque. Elle a permis la séparation de bâtiments mitoyens. La prédiction des distances et celles des étiquettes à trois classes ont permis d'atteindre les mêmes performances, mais chacune avec une architecture différente. On voit ici qu'une étude évaluant chaque type d'étiquette avec plusieurs architectures permettrait de juger de l'intérêt des combinaisons.

La résolution des images d'entrée est un facteur excessivement limitant, même le vainqueur du challenge n'a pu atteindre un F1-score que de 0.255. Nous verrons en Section 3.4 qu'une résolution de 30cm par pixel plutôt que 50cm par pixel permet d'atteindre des performances bien plus élevées.

L'absence de post-traitement a rendu nos modèles peu compétitifs. Bai et Urtasun [2017] ont mis au point un post-traitement utilisant la prédiction du masque pour entraîner un modèle d'apprentissage profond séparant chaque instance d'une même classe. Cette technique pourrait permettre de séparer les bâtiments les uns des autres.

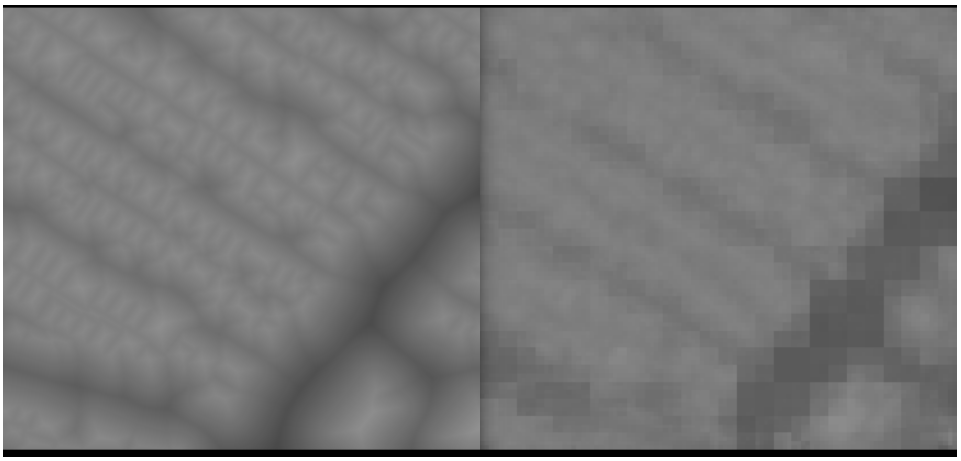

FIGURE 3.19 – Prédiction de la distance avec un modèle SegNet. À gauche le résultat attendu, à droite la prédiction.

La fonction d'erreur utilisée, l'entropie croisée, n'était pas la fonction la plus adaptée. Des fonctions telles que les indices de Dice [Dice, 1945] ou de Jaccard [Jaccard, 1901] sont arithmétiquement bien plus proche de la fonction d'erreur finale et permettent d'obtenir des segmentations dont il est plus aisé d'extraire des polygones. On pourra s'en convaincre facilement en comparant les segmentations obtenues en Figure 3.20. On y voit que l'entropie croisée produit des résultats très flous. Le modèle n'a pas tendance à trancher clairement en faveur d'une classe et beaucoup de pixels sont classifiés en partie comme bâtiment et en partie comme bordure. Avec L'erreur de Dice, les résultats sont très nets, les pixels classifiés clairement dans une classe ou dans une autre, et le modèle apprend à produire des formes consistantes (pas de pixel bordure au milieu de pixels bâtiment).

3.3.4 Conclusion

Nous avons participé à la première édition du challenge SpaceNet dont le but était d'extraire le contour de bâtiments au sein d'images satellites. Nous avons vu que l'architecture SegNet n'était pas assez performante pour concurrencer les meilleurs modèles ; une architecture plus récente serait plus appropriée. Nous avons également vu que le post-traitement jouait un rôle conséquent dans le score final et que la segmentation en 3 classes (intérieur de bâtiment, extérieur de bâtiment et bordure de bâtiment) était probablement la plus efficace. Pour s'en convaincre, une étude comparant plusieurs architectures, essayant de prédire chacune des trois segmentations présentées, combinées à plusieurs post-traitements est nécessaire.

Nos modèles n'ont pas été classés pour cette première édition du challenge SpaceNet. le F1-score calculé après la date limite de soumission, de 0,168, nous aurait valu la 13e place sur les 42 concurrents ayant soumis une solution. Plus de 300 personnes étaient inscrites au challenge mais seules 26 ont obtenu un score non nul, principalement à cause de la quantité de travail à fournir pour obtenir une telle solution et de la courte durée allouée (3 semaines).

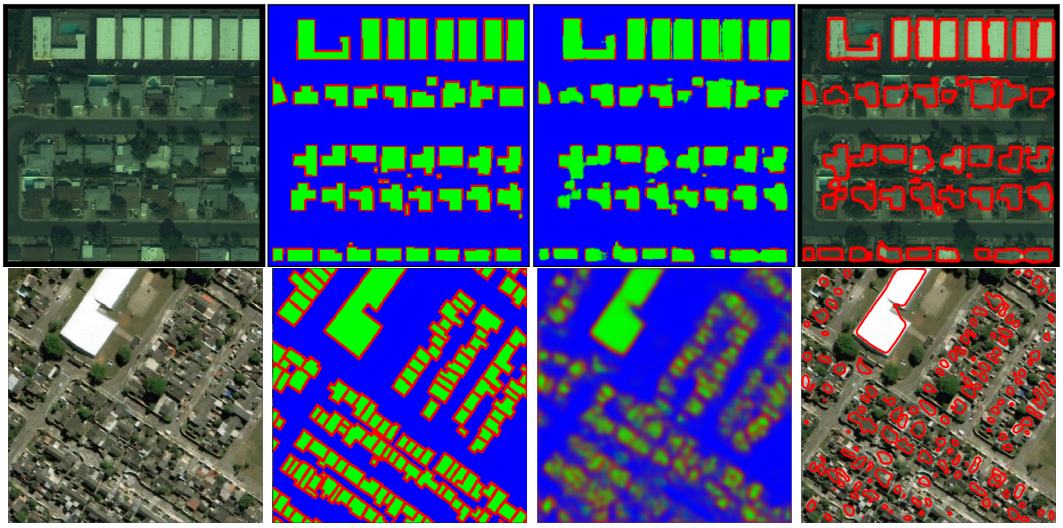

FIGURE 3.20 – Comparaison des résultats en utilisant la fonction d'erreur de Dice (en haut) et l'entropie croisée (en bas). On a ici de gauche à droite l'image d'entrée, l'étiquette, la prédiction, les polygones extraits. La segmentation est bien plus nette et marquée en utilisant l'erreur de Dice.

3.4 Fusion de modèles pour la segmentation de bâtiments

Suite à une seconde édition du SpaceNet challenge, nous étudions la solution gagnante [12] basée sur de la fusion de modèles, afin de l'améliorer.

Cette section décrit donc notre contribution sur la fusion de différentes segmentation de la même image. Elle a fait l'objet d'une publication courte [Delassus et Giot, 2018] au sein du workshop *DeepGlobe building extraction* de la conférence internationale CVPR.

Les modèles de segmentation d'images actuels sont souvent des réseaux neuronaux convolutifs (CNN) popularisés en 2012 par Krizhevsky *et al.* [2012]. Plusieurs modèles peuvent être utilisés simultanément pour segmenter les images, laissant la stratégie de fusion ouverte. Un modèle d'ensemble est un modèle qui fusionne les prédictions de plusieurs sous-modèles et qui fonctionne généralement mieux que le meilleur des sous-modèles fusionnés [Kittler *et al.*, 1998]. Chaque modèle utilisé aura des faiblesses qu'il ne partagera pas forcement avec les autres s'ils sont indépendants. La fusion permet de rectifier les erreurs faites par une minorité des modèles grâce aux résultats des autres modèles. Parmi les stratégies de fusion courantes figure une moyenne non pondérée des prédictions, choix habituel pour la segmentation des images. Cette stratégie a pourtant des vulnérabilités. Par exemple si un des modèles fusionnés fait une erreur tout en étant très sûr de lui, la moyenne s'en trouvera significativement faussée.

12. https://github.com/SpaceNetChallenge/BuildingDetectors_Round2/tree/master/1-XD_XD

Cette étude se concentre sur l'évaluation d'une série de stratégies de fusion afin de trouver l'approche la mieux adaptée à la problématique de détection des bâtiments. Dans cette section, nous comparons une stratégie existante de moyenne simple à des stratégies proposées plus complexes : un modèle de moyenne pondérée (un CNN d'une couche convolutive $1 \times 1 \times 1$), un modèle à moyenne pondérée avec prise en compte du voisinage (un CNN d'une couche convolutive $7 \times 7 \times 1$), et un modèle profond de fusion (une architecture U-Net à 29 couches). Nous nous appuyons sur le challenge SpaceNet [13] qui propose l'apprentissage de modèles pour extraire automatiquement les contours des bâtiments, à partir de leur jeu d'images satellites [14] contenant 10 593 images satellites de 650×650 pixels, de Paris, Vegas, Shanghai et Khartoum avec une résolution de 30cm/pixel.

Cette étude s'est déroulée suite à la deuxième édition du challenge Spacenet. Comme pour la première édition, les meilleurs concurrents ont soumis leur code qui est maintenant accessible sur Github [15]. Nous avons donc étudié la solution gagnante de cette nouvelle édition : un ensemble de trois modèles U-Net [Ronneberger *et al.*, 2015] moyennés. Ces U-Nets ont la même architecture mais ne sont pas entraînés sur les mêmes types d'exemples. Le premier est entraîné sur les couches RGB et MUL redimensionnées, les deux autres sur des blocs extraits des couches MUL pour l'un, MUL + OSM pour l'autre. Ce travail compare cette base de référence, la moyenne non pondérée, à d'autres stratégies de fusion et propose une nouvelle segmentation. Toutes les solutions de fusion utilisent la sortie des trois modèles U-Net précédents comme entrée. Lorsque c'est possible, l'image satellite est également ajoutée à l'entrée. Nous appelons ces modèles des «combineurs». Nous évaluons des combineurs de complexité croissante afin de déterminer des combineurs de faible complexité (super learners [Ju *et al.*, 2017]) sont meilleurs qu'une architecture de CNN récente (combineurs profonds) pour fusionner un ensemble de segmentations.

La principale contribution de cette section est double : (i) la proposition d'une solution de fusion complexe basée sur des méthodes d'apprentissage en profondeur (un CNN utilisant à la fois l'image d'entrée et la sortie des modèles fusionnés) ; (ii) la validation expérimentale de la méthode proposée. La méthode de fusion proposée est comparée à l'originale ainsi qu'à d'autres solutions plus simples que celle proposée. Ces solutions simples sont meilleures que la fusion par moyenne.

Cette section est structurée comme suit. La section 3.4.1 présente les travaux existants sur la fusion des modèles pour la segmentation, ainsi que la solution de fusion ayant remporté la seconde édition du challenge SpaceNet. La section 3.4.2 présente notre contribution ; l'étude de combineurs appliqués au problème de la détection des bâtiments au sein d'images satellites. La section 3.4.3 détaille le protocole expérimental qui valide la contribution. Section 3.4.4 présente les résultats. Les combineurs améliorent la segmentation de l'image ainsi que la détection des bâtiments. La section 3.4.5 conclut ce travail.

13. https://spacenetchallenge.github.io/
14. https://aws.amazon.com/public-datasets/spacenet/
15. https://github.com/SpaceNetChallenge/BuildingDetectors_Round2

3.4.1 État de l'art

Les techniques de fusion pour la segmentation de l'image sont passées en revue, car elles permettent d'établir les limites de la solution de référence avec laquelle nous avons choisi de travailler. On décrit ensuite cette solution qui correspond à la solution gagnante du défi SpaceNet.

3.4.1.1 Apprentissage ensembliste pour la segmentation d'image

La technique consistant à utiliser les sorties de plusieurs modèles et à les combiner pour obtenir une sortie finale est appelée modélisation par fusion ou apprentissage ensembliste. Le but est de parvenir à de meilleures prédictions que celles produites par le plus efficace des modèles fusionnés [Kittler *et al.*, 1998].

Assaad *et al.* [2008] classifient les méthodes de fusion au sein de trois différents groupes : les méthodes de votes, les méthodes par empilement, et les méthodes en cascade.

Les méthodes de votes sont principalement représentées par l'agrégation par amorçage [Breiman, 1996] et le boosting [Schapire, 1990]. La première méthode consiste à créer de nouveaux ensemble d'entraînements en tirant aléatoirement des éléments de l'ensemble d'entraînement d'origine. Différentes copies du modèle, une par nouvel ensemble d'apprentissage, sont entraînées dessus, puis votent à poids égaux pour la prédiction finale. Les forêts aléatoires [Breiman, 2001] sont un exemple typique d'agrégation par amorçage. La seconde méthode consiste à répéter itérativement le processus suivant : concentrer l'apprentissage d'un nouveau modèle sur les exemples pour lesquels le précédent modèle a donné des réponses erronées. Cette méthode a tendance à sur-apprendre. L'algorithme Adaboost [Freund *et al.*, 1996] est un exemple typique de boosting.

Les méthodes par empilement consistent à entraîner un modèle sur les sorties des différents modèles à fusionner. Ceux-ci sont entraînés individuellement, puis les prédictions de chacun des modèles pour un exemple donné sont agrégées et forment alors un exemple pour un nouvel ensemble d'apprentissage. Le modèle final utilise ces exemples pour apprendre à produire une prédiction qui est généralement de meilleure qualité [Wolpert, 1992].

Les méthodes en cascade, appliquée d'abord par Viola et Jones [2001] pour la détection de visage dans des images en temps réel, permettent d'augmenter le jeu de données en produisant de nouvelles propriétés pour chaque exemple. Dans le cas d'une image, on produit de nouvelles couches, concaténées à l'image originale.

Ju *et al.* [2017] montrent que la moyenne non pondérée est une méthode de fusion aussi efficace, sinon meilleure que les méthodes connues. Elle est utilisée par Marmanis *et al.* [2016] sur plusieurs instances de la même architecture de modèle, et par Kamnitsas *et al.* [2017] sur des architectures complémentaires. Elle permet de réduire la variance entre les prévisions, mais est limitée par les modèles donnant des valeurs aberrantes. Le vote majoritaire (utilisé par Dolz *et al.* [2017]) a généralement une performance pixel par pixel inférieure. Il peut cependant produire une «carte d'incertitude», qui peut être utilisée pour faire en sorte que le modèle se concentre dans des secteurs incertains [Wang *et al.*, 2017], ou pour empêcher des classifications ha-

sardeuses de pixels [Kampffmeyer *et al.*, 2016]. Cela pourrait permettre de distinguer les bâtiments proches et améliorer la performance de détection des bâtiments.

La moyenne pondérée avec les poids appris est appelée par Ju *et al.* [2017] un **super learner**. Ce terme est utilisé pour définir un modèle qui prend une segmentation (sortie d'un modèle précédent) comme entrée et en produit une meilleure, ce sont donc des modèles de fusion par empilement. Le super-learner le plus simple peut être vu comme une convolution de 1×1 appliqué sur une entrée de pronfondeur m, le nombre de modèles fusionnés. Le modèle apprend alors les poids optimaux pour le calcul de la moyenne pondérées. Dans leur étude sur l'ensemble de données CIFAR10 [Krizhevsky et Hinton, 2009], ils montrent que ce modèle fonctionne généralement mieux que la moyenne non pondérée. Cependant, ils n'enquêtent pas sur des super-learners plus complexes, comme celui qui fait l'objet de notre étude. De tels super-learners pourraient par exemple prendre en compte le voisinage, ou effectuer des opérations plus complexes en ajoutant plus de couches, résultant en des super-learners profonds.

Le succès des réseaux résiduels [He *et al.*, 2016] montre que l'utilisation combinée de l'entrée et de la sortie d'une couche en entrée de la couche suivante peut conduire à de meilleures performances. Passé à l'échelle du modèle plutôt que de la couche, nous étudions un modèle qui utilisera l'image d'entrée ainsi que les sorties des modèles fusionnés pour apprendre une meilleure segmentation. C'est donc un modèle de fusion par cascade. Cette entrée supplémentaire, l'image d'entrée, place notre modèle en dehors de la catégorie des super-learners. Nous appellerons **combineurs** les modèles combinant la sortie d'autres modèles, avec ou sans entrées supplémentaires.

3.4.1.2 Présentation de la solution de référence.

Solution gagnante du challenge SpaceNet Tous les noms proviennent directement du code soumis afin que le lecteur puisse s'y référer. Quatre sources de données, décrites en détail dans la section 3.4.3, sont utilisées : Images RVB, Images MUL, Images OSM et Vérité terrain.

L'architecture présentée est une fusion de trois modèles différents basés sur U-Net Ronneberger *et al.* [2015] (voir Figure 3.7 pour une architecture détaillée).

- Le U-Net *v9* prend en entrée les images RGB et les images MUL (sans les canaux 0, 3 et 4) **redimensionnées** à 256×256 pixels puis superposées. Les valeurs sont rognées à leur 2e et 98e percentiles et normalisées. Enfin, l'image moyenne est soustraite de chaque image. Le U-Net est ensuite entraîné avec des masques redimensionnés comme étiquettes.
- Le U-Net *v13* prend en entrée des blocs de 256×256 pixels d'images MUL. D'une image de 650×650 pixels 9 blocs de 256×256 pixels sont extraits. La zone de chevauchement des blocs a une largeur de 59 pixels. Les blocs prédits sont combinés pour récupérer des images de 650×650 pixels. On calcule la moyenne des prédictions obtenir une valeur dans les zones de chevauchement de l'image finale.
- Le U-Net *v16* prend en entrée des blocs de 256×256 pixels d'images MUL et OSM superposées. Les blocs prédits sont combinés après prédiction pour récupérer des images de 650×650 pixels. On calcule la moyenne des prédictions

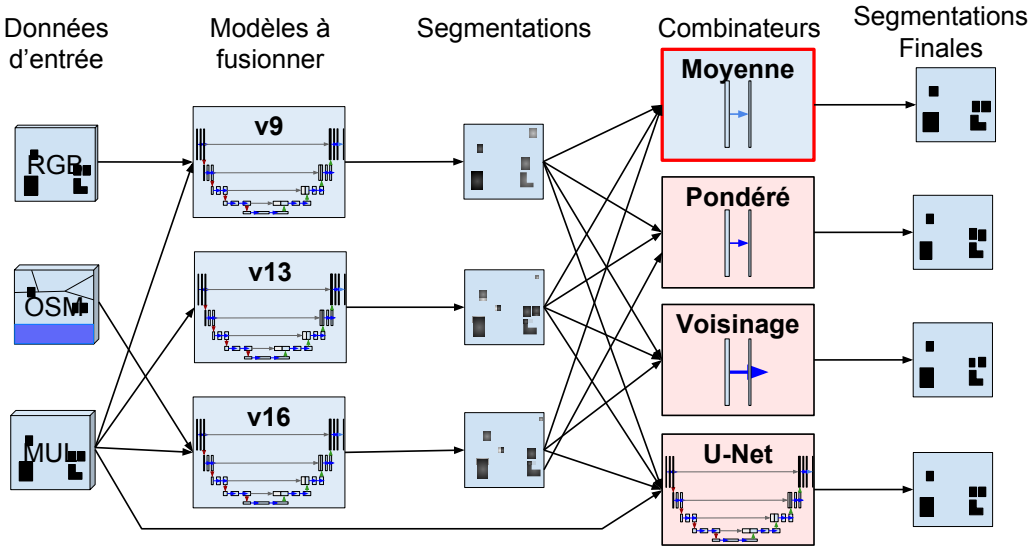

FIGURE 3.21 – Cette représentation de notre processus montre le flux de données passant par les modèles fusionnés qui prédisent les segmentations. Ces segmentations sont utilisées par les combineurs pour produire des segmentations finales. Les polygones sont extraits de ces segmentations. Les cases rouges contiennent les trois combineurs proposés. Dans chaque boîte contenant un modèle, que ce soit un modèle d'entrée ou un combineur, l'architecture est affichée. Une architecture U-Net pour les modèles d'entrée et le combineur U-Net, et une convolution unique pour les autres combineurs.

obtenir une valeur dans les zones de chevauchement de l'image finale.

Pour chaque image satellite, v9 fait une prédiction qui doit être redimensionnée à la taille originale de l'image. Les deux autres modèles, v13 et v16, font 9 prédictions (une sur chaque tranche d'image) qui doivent être combinées ensemble. Ce processus rend difficile la combinaison des trois modèles en un seul modèle pour réaliser la même tâche tout en conservant une architecture similaire. Une fois que la prédiction finale, redimensionnée ou combinée, a été construite pour chaque modèle, un modèle final nommé *v17* prend en entrée les trois prédictions et produit leur moyenne. Ceci correspond à la probabilité pour chaque pixel d'être un pixel de bâtiment. Les probabilités sont ensuite discrétisées à 0 ou 1 en utilisant des arrondis. Afin d'obtenir les contours du bâtiment, on considère qu'un groupe de pixels classés comme bâtiment et connectés entre eux correspond à un bâtiment. Figure 3.21 résume le processus.

Limitations de la solution de référence Plusieurs limites de la solution de référence apparaissent. Premièrement, les modèles sont formés pour prédire un masque, ce qui n'est pas optimal. Lorsque deux bâtiments sont trop proches l'un de l'autre, les contours des bâtiments ne peuvent être extraits du masque correspondant (Figure 3.22) et les deux bâtiments sont considérés comme un seul.

Deuxièmement, le post-traitement est quasi inexistant. Par exemple, on pour-

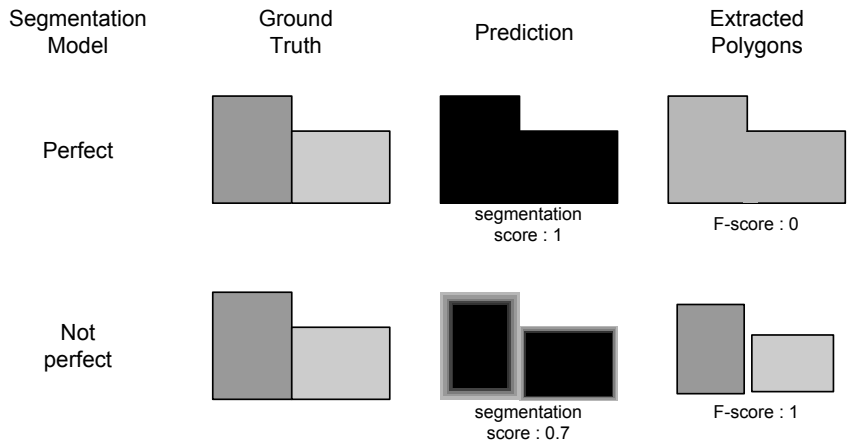

FIGURE 3.22 – Problème de bâtiments adjacents. Lorsque deux bâtiments sont trop proches l'un de l'autre, un modèle parfait (un modèle qui produit exactement les mêmes choses que les étiquettes) produira un seul bâtiment si l'étiquette choisie est un masque. En raison de ce problème, l'amélioration du score de segmentation peut entraîner une diminution du F1-score final.

rait utiliser un modèle de segmentation d'instance [Bai et Urtasun, 2017] ou un algorithme de remplissage par diffusion [Meyer, 1992] pour séparer les bâtiments adjacents. Dans le cas contraire ces bâtiments adjacents seront fusionnés en un seul bâtiment si au moins un pixel adjacent aux deux groupes est classé comme un pixel de bâtiment. Un autre exemple de ce qui peut être fait pendant la phase de post-traitement est le lissage des bords du bâtiment. Les bâtiments ayant la plupart du temps des formes régulières, nous pouvons utiliser ces connaissances pour prédire des formes plus proches de la réalité. Les techniques CRF [Krähenbühl et Koltun, 2011] ont été utilisées pour lisser les bords et améliorer la résolution des segmentations. Cela pourrait aider à préciser les polygones et à les séparer.

Troisièmement, des architectures plus récentes telles que DeepLab [Chen *et al.*, 2016] ou PSPnet Zhao *et al.* [2016] pourraient être utilisées pour essayer d'améliorer le score de segmentation. Ils surpassent l'architecture U-Net dans les tâches de segmentation ImageNet [Russakovsky *et al.*, 2015].

Enfin, une simple moyenne non pondérée est souvent un bon choix par défaut, car elle réduira la variance du résultat. Cependant elle pourrait ne pas être la meilleure solution de fusion, car elle donne la même importance à tous les modèles, quelle que soit leur performance. Elle est donc sujette à des erreurs lorsqu'un des modèles fusionnés est trop optimiste, ou quand une proportion importante des modèles fusionnés est erronée. Cette proportion critique est facilement atteinte lorsque seulement trois modèles sont fusionnés. Nous avons choisi de nous concentrer sur cette dernière limitation.

3.4.2 Combineurs proposés

Ce travail met l'accent sur la fusion de la sortie des trois différentes architectures U-Net, grâce à différentes architectures de combineurs afin de réduire les erreurs de segmentation. Afin de n'évaluer que leur impact, nous n'avons pas modifié le processus d'extraction des bâtiments. Les modèles proposés produisent toujours la probabilité d'être un pixel de bâtiment pour chaque pixel, et l'architecture U-Net des trois modèles fusionnés a été conservée.

Nous avons choisi de concevoir et d'évaluer des modèles d'architectures de complexité croissante. Chacun d'entre eux est implémenté comme un réseau de neurones car c'est une structure qui peut être arbitrairement complexe. Les différentes approches que nous avons utilisées sont : (i) l'approche de la somme pondérée, à peine plus complexe que la méthode de référence ; (ii) l'approche de la somme pondérée avec la prise en compte du voisinage, un super-learner plus complexe ; et (iii) une approche d'apprentissage profond : un combineur profond basé sur U-Net.

L'entrée de ces combineurs est la sortie de plusieurs modèles de segmentation. L'objectif est de combiner ces résultats de segmentation afin de produire leur propre segmentation, supposément meilleure que la meilleure de ses entrées. Des entrées supplémentaires peuvent être données aux combineurs si nécessaire. Figure 3.21, résume ces trois approches.

3.4.2.1 Somme pondérée

L'idée ici est de se débarrasser de la limitation majeure de la moyenne non pondérée. Si l'un des modèles fusionnés a une performance médiocre, nous ne voulons pas que ça pénalise les autres modèles.

Le premier combineur est une somme pondérée des trois segmentations d'entrée, conçue comme un réseau neuronal. Ce réseau neuronal aura trois paramètres à optimiser : un pour chaque entrée qui correspondent aux poids de la somme pondérée. Idéalement, un modèle fusionné plus performant se verra attribuer une pondération plus élevée. Comme expliqué par Ju *et al.* [2017], cela équivaut à une convolution 2D de 1×1. Ce modèle possède une couche cachée unique, le noyau de convolution n'a qu'un pixel par couche d'entrée, il n'y a donc que trois poids à calculer, plus le biais. Cette couche convolutive est représentée dans la Figure 3.21 par une flèche bleue dans la boîte nommée «**Pondéré**». Les poids sont initialisés à 1/3, comme une moyenne non pondérée, avec biais nul. De cette façon, le modèle peut directement commencer à réaliser une meilleure segmentation que la moyenne actuelle. La fonction d'activation utilisée est une activation de rognage, appliquée pour éviter les prédictions en dehors de $[0; 1]$:

$$\sigma(x) = \begin{cases} 0, & \text{si } x \leq 0 \\ 1, & \text{si } x \geq 1 \\ x, & \text{sinon} \end{cases} \tag{3.6}$$

Elle compense le fait qu'un modèle linéaire sans activation sigmoïde n'a pas de moyen pour contraindre ses prédictions à rester dans l'intervalle désiré.

Nous appelons ce super-learner l'approche du combineur *Pondéré*.

3.4.2.2 Somme pondérée prenant en compte le voisinage

L'idée ici est d'améliorer la somme pondérée en donnant au combineur la possibilité de réduire le bruit et de lisser les bords des bâtiments grâce à la connaissance des pixels voisins.

Le second combineur est une somme pondérée avec la prise en compte du voisinage, conçue sous la forme d'un réseau neuronal. Chaque pixel de la segmentation produite sera le résultat de la somme pondérée des pixels d'entrée correspondants ainsi que des voisins de ces pixels. On cherche ici à utiliser les classifications voisines pour améliorer la classification calculée, comme dans toute approche convolutive. Si tous les voisins sont classés comme bâtiments, la probabilité de classer le pixel comme bâtiment peut être plus élevée. Il est également possible de lisser les bords ou de réduire le bruit.

La somme pondérée avec l'approche de la connaissance du voisinage correspond à une convolution $N \times N$ 2D. C'est la seule couche cachée et elle utilise la même fonction d'activation (équation (3.6)) que la somme pondérée. Elle est également initialisée de telle sorte que la moyenne non pondérée soit calculée. Cette couche convolutive est représentée dans la Figure 3.21 par une flèche bleue dans la boite nommée **Voisinage**. Ce modèle a 148 paramètres à apprendre. 7×7 pour chacun des trois canaux et un paramètre de biais. Ce combineur est censé fonctionner au moins aussi bien que la solution de référence, car l'état de la somme pondérée simple peut être atteint en mettant à 0 chaque poids autre que celui qui se trouve au centre de chaque fenêtre de convolution. Nous nommons cette variante l'approche de combineur *Voisinage*.

3.4.2.3 Combineur profond

L'idée ici est de calculer les résidus des modèles fusionnés en utilisant à la fois les segmentations prévues et l'image d'entrée.

L'approche par un combineur profond vise à améliorer la segmentation en permettant au réseau neuronal profond de calculer des caractéristiques complexes telles que la détection des contours, la distance par rapport aux pixels du bâtiment, etc. De plus, ce combineur utilise non seulement les trois segmentations, mais aussi l'image

TABLE 3.6 – Comparaisons des combineurs proposés sur leurs entrées, nombre de couches cachées et paramètres à apprendre.

Combineur	**Entrée**	**Couches cachées**	**Paramètres**
Référence [16]	segmentations de v9, v13 & v16	1	0
Pondéré	segmentations de v9, v13 & v16	1	4
Voisinage	segmentations de v9, v13 & v16	1	148
U-Net	segmentations de v9, v13 & v16 + MUL	29	7 838 273

originale en entrée. Cette image peut être utilisée par le combineur pour déterminer les résidus des modèles d'entrée et corriger leurs erreurs. Nous utilisons les canaux MUL de l'image originale comme entrée. Les modèles entraînés sur les canaux MUL donnent de meilleurs résultats que ceux entraînés sur les canaux RGB. Utiliser ces deux groupes de canaux ainsi que les canaux OSM donne des résultats encore meilleurs. Cependant, la taille des images (656×656) limite le nombre de canaux utilisables, sous peine de dépasser (en incluant les poids du modèles) l'espace mémoire de la carte graphique.

Nous utilisons la même architecture U-Net que les réseaux fusionnés, décrite dans la Figure 3.7, avec 29 couches cachées et 7 838 273 paramètres. Cette architecture est également représentée dans Figure 3.21 par la boîte nommée **U-Net**.

Cette architecture est généralement utilisée pour segmenter les images. Combiner des segmentations en une seule peut être considéré comme effectuer une nouvelle segmentation sur une image dont les canaux sont une superposition des segmentations existantes. Il est donc logique d'utiliser une architecture de segmentation pour réaliser la fusion.

Ce combineur n'a pas été initialisé manuellement pour effectuer la moyenne non pondérée de la segmentation précédente, car cela compromettrait le processus d'apprentissage. Les poids sont initialisés avec une distribution uniforme entre -0,05 et +0,05. Nous nommons cette variante l'approche du combineur *U-Net*.

Les propriétés des combineurs sont résumées dans la table 3.6. Les combineurs avec plus de paramètres sont plus complexes. On s'attend à ce qu'ils soient plus difficiles à optimiser, mais aussi à obtenir de meilleurs résultats. Ces trois combineurs

16. https://github.com/SpaceNetChallenge/BuildingDetectors_Round2/tree/master/1-XD_XD

ont été évalués selon le protocole décrit dans la section suivante.

3.4.3 Protocole expérimental

Cette section présente toutes les informations nécessaires pour reproduire notre travail.

3.4.3.1 Jeu de données

Cette étude utilise les données du second challenge SpaceNet, vues en Section 3.2.1

Chaque combineur est évalué sur les 4 villes de la base de données par rapport au modèle de référence, appelé approche *Référence* (voir 3.4.1.2). Pour chaque ville, 30% des images sont sélectionnées au hasard pour le processus de validation. Les 70% restantes sont augmentées par des rotations (90, 180 et 270 degrés) et des retournements verticaux et horizontaux. Ces augmentations sont là pour augmenter artificiellement la taille de l'ensemble de données, car quelques centaines d'images suffisent rarement à entraîner un modèle d'apprentissage profond. Le modèle est entraîné avec ce dernier ensemble, l'ensemble d'entraînement. L'ensemble de validation n'est pas visible pendant l'entraînement. À la fin de l'entraînement, nous sélectionnons l'époque la plus performante par rapport à l'ensemble de validation. Toutes les performances rapportées sont calculées sur l'ensemble de validation. Il aurait été préférable de rendre compte des performances sur un troisième ensemble de données, l'ensemble de test, qui n'aurait pas été utilisé dans le processus de sélection de la meilleure époque, afin d'éviter tout sur-apprentissage. Cependant, le nombre d'images était trop petit pour être réduit encore plus (voir Tableau 3.7). Un bon indicateur du fait qu'il n'y a pas eu de sur-apprentissage est que notre solution a obtenu un F1-score proche de nos évaluations lors du workshop Deepglobe CVPR 2018 sur la détection des bâtiments [17], évalué sur des images jamais vues auparavant. Ce score nous a permis de nous placer sixième au classement final du workshop, ayant donné lieu à une présentation sous forme de poster à la conférence CVPR [Delassus et Giot, 2018].

Les quatre villes représentées dans le jeu de données sont très différentes les unes des autres. Cela nous permet de vérifier que notre solution peut être appliquée dans diverses situations. Elles se situent chacune sur un continent différent et se différencient par les couleurs, la taille des bâtiments, les disparités de construction (les bâtiments de Vegas sont tous très similaires les uns aux autres par exemple), l'organisation dans l'image (les bâtiments de Khartoum sont principalement en petits groupes, ceux de Paris sont plus éparses...)

3.4.3.2 Métriques d'évaluations

Les métriques présentées ici sont utilisées pour comparer nos combineurs à l'existant. Deux d'entre elles sont utilisées pour évaluer chaque modèle : le coefficient de Jaccard [Jaccard, 1901] et le F1-score. Le premier évalue la segmentation, tandis que

17. https://competitions.codalab.org/competitions/18544

TABLE 3.7 – Taille des ensembles d'apprentissage et de validation pour chaque ville

Ville	**# Images d'apprentissage**	**# Images de validation**
Vegas	2 695	1 156
Paris	803	345
Shanghai	3 207	1 375
Khartoum	708	304

le second évalue la détection des bâtiments. Le coefficient de Jaccard est la mesure la plus proche du F-score lorsqu'il s'agit de comparer des images. Ils sont tous deux corrélés à la proportion de bonnes classifications et anti-corrélés au nombre d'erreurs de classification. On a vu dans l'étude précédente que ce type de fonction de coût mène à de meilleures performances que l'entropie croisée.

La segmentation n'est que la première étape du processus, mais les études ne vont souvent pas plus loin. Le score de segmentation doit être défini car c'est celui que le modèle de segmentation tente d'optimiser. La deuxième étape consiste à utiliser la segmentation produite pour extraire les contours des bâtiments. Le F1-score est utilisé pour évaluer les performances de la détection de bâtiments. Compte tenu d'un étiquetage de la vérité terrain y et d'une segmentation prédite à évaluer $\hat{y}$, le coefficient de Jaccard est défini ainsi :

$$J(y, \hat{y}) = \frac{y \cap \hat{y}}{y \cup \hat{y}} \tag{3.7}$$

Appliqué à une image, ceci devient :

$$J(y, \hat{y}) = \frac{\sum (y * \hat{y})}{\sum (y + \hat{y}) - \sum (y * \hat{y})} \tag{3.8}$$

En supposant que $J(0, 0) = 1$. Comme le coefficient de Jaccard produit une valeur entre 0 et 1, il est facile de le transformer en fonction de coût :

$$J_l(y, \hat{y}) = 1 - J(y, \hat{y}) \tag{3.9}$$

Le F1-score est calculé comme présenté précédemment en Section 3.2.2.

3.4.3.3 Optimisation des paramètres

Afin de maximiser l'indice Jaccard moyen, nous utilisons l'optimiseur Adam [Kingma et Ba, 2014]. Le taux d'apprentissage initial est fixé à $1e-3$ pour la somme pondérée, $1e-4$ pour l'approche Voisinage, et $5e-5$ pour l'approche U-Net. Ce sont pour

chacune de ces trois méthodes les taux d'apprentissage les plus élevés ne menant pas à une divergence lors de l'apprentissage. À la 10e époque, le taux d'apprentissage est divisé par 5 car la publication originale de l'optimiseur Adam suggère une décroissance exponentielle du taux d'apprentissage. En le conservant à sa valeur de départ nous observons régulièrement une divergence de l'apprentissage après la 10e époque. Nous entraînons chaque modèle pour 20 époques. En effet, avec cette configuration on observe empiriquement une convergence de l'apprentissage autour de la 15e époque, rarement plus, et jamais après la 20e. Nous calculons ensuite le F1-score et l'indice de Jaccard sur l'ensemble de validation pour chacune d'entre elles. Les modèles sont entraînés ville par ville, et évalués sur la même ville que celle ayant servit à l'apprentissage.

3.4.3.4 Matériel

Les expériences ont été effectuées sur un ordinateur Linux avec les caractéristiques suivantes : Processeur Intel(R) Core(TM) i7-6950X à 3.00GHz, trois GPU Titan X Pascal (utilisés pour entraîner plusieurs modèles en parallèle, avec un modèle par GPU) et 120Go de RAM, utilisant Keras [Chollet *et al.*, 2015] 1.2.2 avec Tensorflow [Abadi *et al.*, 2016] 1.4.0.

3.4.3.5 Hyper-paramètres

Compte tenu de la quantité de mémoire GPU à notre disposition et du fait que nous ne réduisons pas la taille d'entrée à 256x256, les tailles de batch étaient maintenues à la taille par défaut de 32 pour les deux modèles légers (Pondéré et Voisinage) mais ont dû être réduites à 1 pour l'architecture U-Net.

Nous avons fixé la taille de convolution du modèle Voisinage à 7 pixels. Cela correspond à 2,1 mètres, ce qui est généralement assez grand pour avoir deux bâtiments proches dans le champ réceptif.

Un seuil de surface prédéfini, en pixels, permet d'exclure les bâtiments trop petits lors du post-traitement. Tous les bâtiments prédits sont vérifiés un à un, ceux étant trop petits sont supprimés. Les bâtiments de moins de 20 pixels de surface n'ont pas été étiquetés. Cela aide à réduire le bruit. Le seuil est initialisé avec le seuil dans $\{30, 60, 90, 120, 150, 180, 210\}$ donnant le meilleur score pour le combineur de référence. Pour comparer deux modèles, nous entraînons chaque modèle pendant 20 époques, puis nous sélectionnons l'époque qui produit le meilleur F1-score avec le seuil prédéfini. Une fois l'époque sélectionnée, nous sélectionnons le meilleur seuil pour cette époque.

La section suivante présente les résultats obtenus selon ce protocole.

3.4.4 Résultats

Cette section présente les performances de nos combineurs, les visualisations des bâtiments détectés et se termine par une discussion sur nos résultats.

TABLE 3.8 – Meilleur coefficient de Jaccard, et gain par rapport à la méthode de référence, pour chaque ville et chaque modèle sur les images de validation.

Ville	**Référence**	**Pondéré**	**Voisinage**	**U-Net**
Vegas	0,789 0 -	0,796 6 +0,96%	0,814 8 +3,27%	**0,841 7** **+6,68%**
Paris	0,642 6 -	**0,795 9** **+23,86%**	0,792 0 +23,25%	0,737 8 +14,81%
Shanghai	0,657 0 -	0,691 1 +5,19%	0,705 4 +7,37%	**0,878 7** **+33,74%**
Khartoum	0,689 8 -	0,722 1 +4,68%	0,752 4 +9,07%	**0,768 8** **+11,45%**

TABLE 3.9 – Meilleur F1-score, et gain par rapport à la méthode de référence, pour chaque ville et chaque modèle sur les images de validation.

Ville	**Référence**	**Pondéré**	**Voisinage**	**U-Net**
Vegas	0,855 9 -	0,863 9 +0,93%	0,859 4 +0,41%	**0,863 9** **+0,93%**
Paris	0,680 5 -	0,678 8 -0,25%	0,689 4 +1,31%	**0,708 0** **+4,04%**
Shanghai	0,562 7 -	0,564 6 +0,34%	0,575 9 +2,35%	**0,579 4** **+2,97%**
Khartoum	0,585 5 -	0,628 4 +7,33%	**0,644 3** **+10,04%**	0,629 0 +7,43%

3.4.4.1 Performances

Le tableau 3.8 montre que les combineurs ont pu améliorer la segmentation pour toutes les villes par rapport à la solution de référence. Dans ce tableau, nous présentons la valeur maximale du coefficient de Jaccard obtenue pour chaque ville avec chaque modèle sur l'ensemble de données de validation. Le meilleur modèle est mis en évidence pour chaque ville. La deuxième ligne indique le gain par rapport à la ligne de base. Le gain est défini comme $Gain = (Nouveau_Score - Reference)/Reference$.

La référence a déjà de très bonnes performances de segmentation sur l'ensemble de données de Vegas et est donc plus difficile à améliorer. Cependant, elle a de mauvaises performances de segmentation sur les autres ensembles de données, et nous avons été en mesure de l'améliorer de manière significative : les performances pour Shanghai sont supérieures de plus de 33% à celles de la référence. Dans l'ensemble, l'approche U-Net permet d'améliorer la segmentation. L'écart important entre la performance de la segmentation entre la méthode de référence et la somme pondérée de le jeu de données de Paris indique qu'au moins l'un des modèles fusionnés fonctionne mal.

Le tableau 3.9 montre que l'amélioration de la segmentation ne se traduit pas très bien en amélioration de la détection des bâtiments. Dans ce tableau, nous présentons la valeur maximale du F1-score obtenue pour chaque ville avec chaque modèle sur l'ensemble de données de validation. Le meilleur modèle est mis en évidence pour chaque ville. Le gain est défini comme précédemment. Les combineurs ont été plus performant que la méthode de référence pour tous les jeux de données, à l'exception de la moyenne pondérée sur Paris, dont la performance a été très légèrement inférieure. La meilleure approche globale est celle de U-Net. Comme nous l'avons dit, la méthode de référence était déjà très performante à Vegas par rapport à d'autres villes et n'a été que légèrement améliorée. L'approche pondérée est beaucoup plus simple que l'approche U-Net et a donné des résultats similaires pour Vegas et Khartoum, mais elle ne fonctionne pas aussi bien pour Paris et Shanghai.

3.4.4.2 Visualisation des bâtiments détectés

La Figure 3.23 et la Figure 3.24 montrent une sélection de résultats de l'approche U-Net. Ces images ont été produites à l'aide du Visualiseur de détecteur de bâtiment SpaceNet Challenge [18]. La vérité terrain et les polygones prédits sont superposés sur les images RGB. La vérité terrain et les polygones prédits correspondants sont blancs. Les polygones jaunes sont des faux positifs (polygone prédit sans vérité terrain correspondante). Les polygones bleus sont des faux négatifs (polygone de vérité terrain sans prédiction correspondante).

La Figure 3.23 montre les erreurs faites par le combineur U-Net. La fusion de bâtiments adjacents est une erreur très fréquente, que l'on trouve dans toutes les villes et à laquelle il faut remédier. La deuxième image révèle une difficulté possible à détecter les bâtiments présentant un écart d'échelle significatif à Shanghai. Une autre explication possible est que les toits blancs ne sont en fait pas des bâtiments mais des abris pour les véhicules du parking, ou n'ont pas été étiquetés comme bâtiments dans d'autres images. La troisième ligne montre une image de Khartoum. À droite, les polygones prédits sont affichés. On peut voir sur l'image de multiples formes rectangulaires, plus petites que les bâtiments environnants. Sur la gauche, certains d'entre eux sont surlignés en bleu, ce qui signifie qu'il s'agit de bâtiments que le combineur a manqués. Cependant, d'autres formes identiques ne sont pas mises en évidence et ne sont donc pas considérées comme des bâtiments. Cette incohérence dans l'étiquetage se retrouve régulièrement dans toutes les villes.

18. https://github.com/SpaceNetChallenge/BuildingDetectorVisualizer

La figure 3.24 montre que le combineur a pu corriger certaines erreurs trouvées dans les étiquettes. Ces résultats sont encourageants puisque le principal cas d'utilisation de cette étude serait de détecter automatiquement les erreurs dans un ensemble de données géographiques ouvertes comme OpenStreetMap [19].

FIGURE 3.23 – Exemple d'erreurs du combineur U-Net. Dans la première image, les bâtiments adjacents sont fusionnés parce qu'ils étaient trop proches les uns des autres. Dans la deuxième image, seuls les grands bâtiments à plusieurs étages ont été détectés, alors que les petits toits blancs ne l'ont pas été. Le troisième cas montre des polygones prédits à droite, et met en évidence les bâtiments manqués à gauche.

3.4.4.3 Discussion

Nous avons pu mettre en évidence deux sources principales d'erreurs. Le premier est le problème des bâtiments adjacents (Figure 3.22). Lorsque nous réduisons l'in-

19. https://www.openstreetmap.org

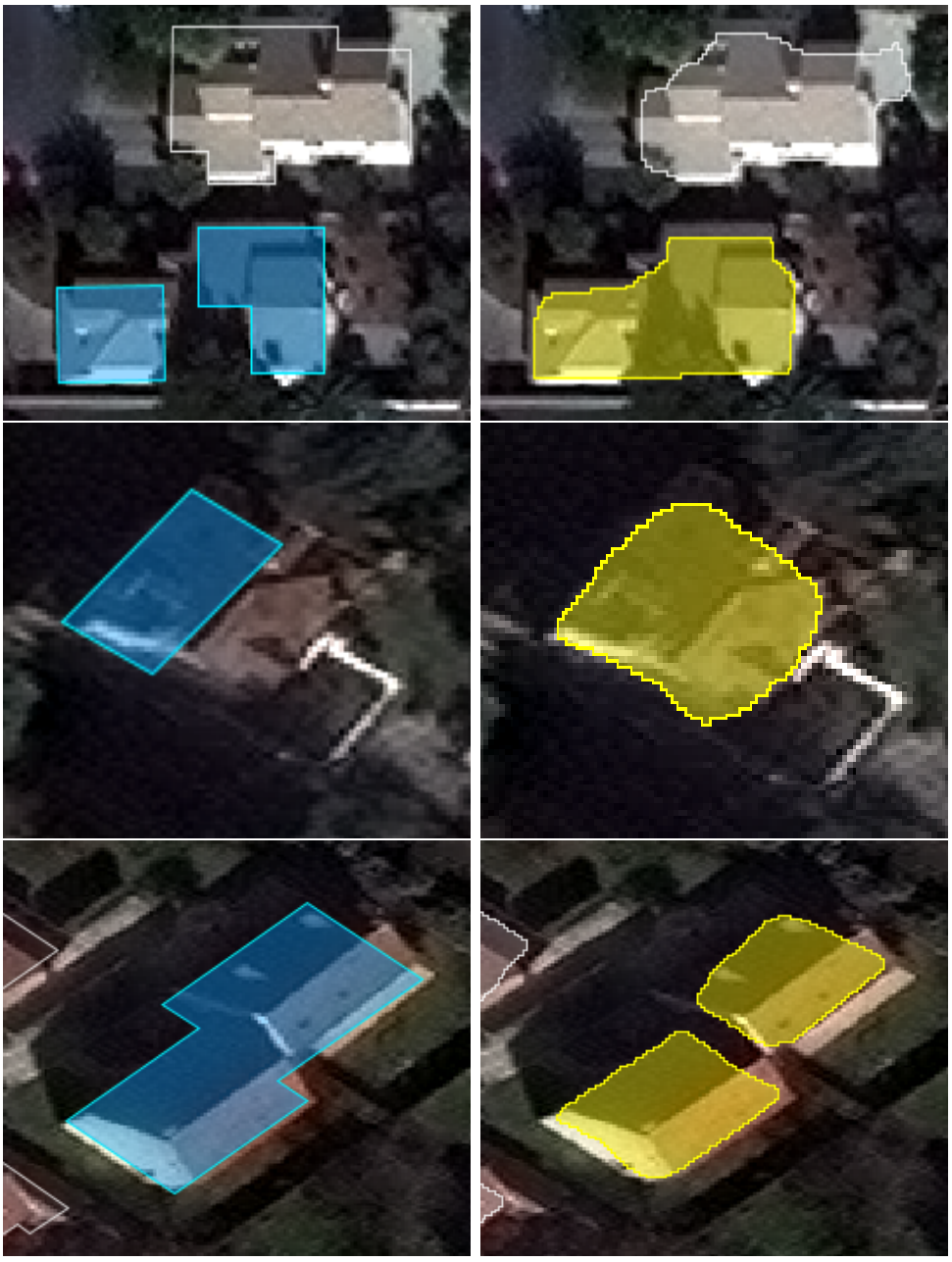

FIGURE 3.24 – Exemple de cas où la vérité terrain (à gauche) est fausse, mais le combineur U-Net a prédit des polygones corrects malgré la pénalisation lors de l'apprentissage. Dans le premier cas, un arbre cache la partie centrale du bâtiment qui a été incorrectement étiqueté comme deux bâtiments différents. Le deuxième cas est une erreur d'étiquetage sans explication. Dans la troisième image, deux bâtiments ont été incorrectement étiquetés comme un seul, mais le combineur a réussi à les séparer.

FIGURE 3.25 – Exemple d'image pour chaque ville. Ces images ont été choisies en raison de la densité élevée en bâtiments. De gauche à droite puis de haut en bas, les villes sont : Vegas, Paris, Shanghai et Khartoum.

certitude du modèle autour des bâtiments, les bâtiments adjacents sont considérés comme un seul bâtiment au lieu de deux bâtiments différents. La seconde provient de nombreuses erreurs dans les données étiquetées. Ces erreurs peuvent empêcher les modèles d'apprendre des fonctions utiles ou les punir lorsqu'ils le font.

Nous constatons que malgré une très bonne amélioration de la segmentation (>33%) à Shanghai, le score de détection n'a été amélioré que de 3%. L'ensemble de données de Shanghai a beaucoup de très petits toits qui sont parfois classés comme des bâtiments et parfois non. D'une part, les petits bâtiments ne représentent pas beaucoup de pixels dans l'image par rapport aux grands. Un modèle entraîné avec un score de segmentation donnera la priorité à la segmentation des grands bâtiments au prix de la qualité de segmentation des petits bâtiments. D'autre part, le F1-score n'est pas influencé par la taille des bâtiments : un faux négatif aura le même impact

si un grand bâtiment ou un petit bâtiment a été manqué. Ainsi, le fait de manquer un grand nombre de petits bâtiments aura un impact énorme sur le F1-score, tout en ne diminuant pas significativement l'indice de Jaccard. Ce problème vient du fait que nous utilisons une fonction de coût «écran», optimisant la segmentation pendant la phase d'apprentissage et non la détection elle-même. Cependant, nous ne connaissons pas de solution permettant de former un CNN directement sur le score de détection. Pour résoudre ce problème, on pourrait fusionner des modèles chargés de segmenter des bâtiments de différentes tailles. Un modèle dédié à la segmentation de bâtiments de petite taille n'aura pas d'intérêt à favoriser la segmentation de grands bâtiments. Cependant, si seule une partie des bâtiments sont étiquetés pour chacun de ces modèles, une adaptation de la fonction d'erreur peut être nécessaire (voir Section 3.5).

Pour résoudre le problème des bâtiments adjacents, la prédiction de trois classes : bâtiment, bordure, et fond peut s'avérer utile, comme vu Section 3.3. Il est alors plus facile de diviser les bâtiments adjacents. Yuan [2017] utilise un champ distance comme étiquettes. Il est alors possible de configurer le post-traitement et de définir une distance prédite (0 dans le masque) comme bordure. Mais il faut un grand champ réceptif pour traiter ce jeu de données, nos premiers travaux utilisant cette technique n'ont pas été concluants. Bai et Urtasun [2017] suggèrent que l'apprentissage d'une distance est une tâche trop complexe et introduisent une étape intermédiaire, en prédisant d'abord la direction de la bordure la plus proche de l'image d'entrée et sa segmentation prédite (la même entrée que notre approche U-Net) et en utilisant ensuite les directions prédites pour prédire la distance. Nous avons également mis en œuvre cette solution, elle n'a pas été en mesure de séparer les bâtiments adjacents. Le modèle a tendance à apprendre la distance jusqu'à la zone de fond la plus proche au lieu de la bordure la plus proche, car les polygones de vérité terrain sont presque toujours séparés par au moins un pixel. Nos travaux futurs se concentreront sur d'autres solutions de segmentation d'instances afin de résoudre le problème des bâtiments adjacents. En effet, la séparation des bâtiments doit être améliorée avant de progresser davantage dans la segmentation.

Nous avons utilisé la même architecture pour le combineur U-Net que celle utilisée pour les modèles combinés. Davantage d'architectures devraient être testées pour déterminer si les CNNs classiques sont les meilleurs combineurs. Maggiori *et al.* [2017] ont utilisé un réseau neuronal récurrent pour améliorer leur segmentation, offrant ainsi une solution alternative aux CNNs.

Le combineur profond a d'abord été testé avec la segmentation prédite combinée à l'entrée RGB. Nous avons remplacé l'entrée RGB par des entrées MUL et obtenu de bien meilleurs résultats (Pas d'amélioration avec les images RGB et jusqu'à 10% d'amélioration du F1-score avec les images MUL). L'utilisation simultanée des entrées RGB, MUL et OSM pourrait conduire à des résultats encore meilleurs, mais nécessiterait une réduction de la taille de l'image résultante, car la mémoire du GPU ne pourrait contenir le modèle et une image si grosse. Nous savons déjà que le meilleur des trois modèles combinés utilise à la fois les entrées MUL et OSM, ce qui indique une amélioration lors de l'utilisation des données OSM.

3.4.5 Conclusion

Cette section traite du problème de l'extraction de contours de bâtiments à partir d'images aériennes. Nous y proposons une solution de fusion pour un ensemble de modèles U-Net utilisés pour segmenter les images aériennes et extraire les contours des bâtiments de la segmentation. La solution proposée conserve l'architecture U-Net utilisée par les modèles fusionnés. Elle produit une segmentation à l'aide de trois segmentations prédites par les modèles fusionnés et des couches MUL de l'image correspondante.

Nous avons utilisé la solution gagnante du challenge SpaceNet, une fusion moyenne non pondérée, comme référence pour évaluer les avantages du combineur proposé, ainsi que deux super-learners : une somme pondérée et une somme pondérée des pixels voisins. Notre solution a systématiquement donné de meilleurs résultats que les modèles existants, avec une amélioration de l'indice de Jaccard allant jusqu'à 33% et une amélioration du F1-Score allant jusqu'à 10% par rapport à la référence. L'amélioration du F1-score n'était pas proportionnelle à l'amélioration de l'indice Jaccard, indiquant que les travaux futurs devraient se concentrer sur la séparation des bâtiments adjacents, au moyen d'une meilleure étiquette telle que l'ajout d'une classe *bordure de bâtiment* ou par un post-traitement de la segmentation, comme proposé par Bai et Urtasun [2017].

Nous avons vu que notre méthode de fusion peut améliorer significativement les scores de segmentation dans le contexte de la détection des bâtiments. ImageNet est généralement utilisé pour comparer les modèles de segmentation. Il serait intéressant d'évaluer notre solution sur cet ensemble de données en fusionnant plusieurs des soumissions les plus performantes sur ce jeu de données.

Nous avons vu que les étiquettes contiennent quelques erreurs, et que le modèle a pu corriger certaines d'entre elles. Il serait intéressant d'étudier l'impact du bruit contenu dans les étiquettes, et les manières utilisables pour rendre un modèle résistant à ce bruit.

3.5 Analyse de la résistance au bruit

La finalité de ce travail de recherche est d'améliorer les données provenant de sources ouvertes telles qu'OpenStreetMap (OSM) en corrigeant les variations par rapport à la réalité décrite par les images satellites, appelées le bruit. Pour y parvenir, nos modèles apprennent à déterminer les contours des objets cartographiés, depuis des images satellites. On se restreint ici à la détection des bâtiments au sein de ces images.

On considère que le nombre d'erreurs dans les étiquettes de notre jeu de données est négligeable. Cependant, dans une application réelle, les polygones de la vérité terrain ne sont pas parfaits. Dans le cas d'OSM, ils sont faits par une multitude de contributeurs, certaines zones ne sont pas du tout cartographiées, les données ne sont pas forcément mises à jour, etc. C'est ce qui motive nos travaux. En effet, si l'on souhaite corriger les données d'OSM, c'est à partir de ces mêmes données provenant d'OSM que l'apprentissage sera fait. Ce sont les mêmes données qui doivent être

corrigées et qui servent de vérité terrain. On suppose dans ce cas là que les étiquettes sont bruitées.

Se posent alors les questions suivantes : est-il possible d'apprendre à partir d'un jeu de données dont la vérité terrain comporte des erreurs ? À quel point ces erreurs influencent-elles l'apprentissage ? Les modèles obtenus sont-ils assez bons pour produire une prédiction débarrassée de ces erreurs, afin de pouvoir les corriger ?

Les types de bruits trouvés au sein des données OSM ont été empiriquement décelés lors de nos propres contributions à l'amélioration des données OSM. Les **translations** sont très communes, souvent expliquées par l'angle selon lequel l'image satellite a été capturée. La **fusion** de bâtiments est une erreur consistant à considérer plusieurs bâtiments comme un unique bâtiment, causé par une grande proximité entre eux. À l'inverse, la **séparation** est l'erreur consistant à découper un bâtiment en plusieurs bâtiments. Cela peut être dû à la luminosité changeante, un objet dissimulant une partie du bâtiment (arbre, poteau...), ou être une erreur inexpliquée. La **simplification** intervient lorsque des bâtiments aux formes complexes sont décrits par des polygones trop simples, ou lorsque des bâtiments de forme concaves sont décrits par des polygones convexes. Elle peut-être simulée par le remplacement d'un polygone par son enveloppe convexe. L'**ajout** ou la **suppression** de bâtiments peut s'expliquer par une acquisition de données satellites non synchrone à l'acquisition de données décrivant les bâtiments. Toutes ces erreurs sont également expliquées par le facteur humain. En l'absence de méthodes automatiques de cartographie, cette tâche est soit semi-automatisée, des humains venant corriger les erreurs créées, soit complètement manuelle comme pour OSM. D'autres transformations telles que des rotations ou des homothéties ne sont pas des erreurs communément rencontrées au sein des données manipulées. Nous verrons en Section 3.5.3.1 les raisons pour lesquelles nous limitons cette étude aux bruits de suppressions.

3.5.1 État de l'art

Cette section présente les travaux existants sur l'étude de classification tenant compte de bruit dans les étiquettes.

Frénay et Verleysen [2014] introduisent une taxonomie des bruits au sein des étiquettes pour les problèmes de classification. Dans le cas d'une segmentation, la classification se fait à l'échelle du pixel ; à chaque pixel est attribué une classe, dans notre cas *bâtiment* ou *fond*. On distingue pour chaque exemple sa classe véritable (ce pixel est il véritablement un pixel de classe bâtiment ou de classe fond) et l'étiquette associée (dans la vérité terrain, la classe attribuée au pixel est elle bâtiment ou fond). Lorsque l'étiquette (de la vérité terrain) associée à l'exemple n'est pas identique à la classe véritable, on parle d'étiquette **bruitée**. Seuls les bruits stochastiques sont étudiés, on ne s'intéresse pas au bruitage malicieux ou antagoniste (qui cherche délibérément à exploiter une faille du modèle, comme Szegedy *et al.* [2013]). Selon leur taxonomie, il existe trois types de bruits : *Noisy Completely at Random Model* (CARM), qui suppose que la probabilité d'erreur d'étiquetage ne dépend ni de l'étiquette réelle, ni de l'étiquette observée, ni de l'entrée ; *Noisy At Random* (NAR) qui suppose que la probabilité qu'une étiquette ait subi une transformation dépend de sa véritable classe ; *Noisy Not at Random Model* (NNRM) qui suppose

que le bruit arrive pour des raisons rationnelles, telle que la similitude de classes proches (deux oiseaux de race différente mais de même couleur ont plus de chance d'être confondus qu'une voiture et un chat) et dépend à la fois de la classe réelle, de l'étiquette observée, et de l'entrée donnée. Notre problème de bruit appartient à la catégorie NAR. En effet, nous nous intéressons aux bâtiments non étiquetés, donc seuls les étiquettes de pixels dont la véritable classe est «bâtiment» sont altérés par ce bruit. Ils montrent qu'un tel bruit peut avoir des conséquences désastreuses sur les performances de certain classificateurs tels que adaboost [Freund *et al.*, 1999], ou SVM [Cortes et Vapnik, 1995], alors que d'autres tels que les Bayes naïfs conservent de bonnes performances tant que le bruit n'est pas prépondérant. La segmentation à l'aide de réseaux de neurones n'est pas étudiée.

Mnih et Hinton [2012] souhaitent surmonter la difficulté d'entraîner un modèle à segmenter les routes depuis des images à l'aide de données OSM contenant des étiquettes bruitées. Les deux types de bruits qu'ils rencontrent dans leurs étiquettes sont des omissions (les routes ne sont pas cartographiées), et des erreurs de localisations (les routes sont présentes mais mal placées dans l'image). Pour lutter contre les omissions, ils modifient leur fonction de coût de manière à moins pénaliser les faux positifs. En effet, une forte pénalisation des faux positifs, alors que l'on sait que le modèle avait potentiellement raison, entraînerait un modèle moins sûr de lui qui produirait plus de faux négatifs. Dans cet article, on appelle m_i la véritable classe du pixel i (0 pour le fond, 1 pour le bâtiment), $\tilde{m}_i$ l'étiquette du pixel i et $\hat{m}_i$ la prédiction pour le pixel i. On définit θ_0 et θ_1 comme la proportion de faux positifs et de faux négatifs au sein des labels :

$$\begin{aligned} \theta_0 &= p(\tilde{m}_i = 1 | m_i = 0) \\ \theta_1 &= p(\tilde{m}_i = 0 | m_i = 1) \end{aligned} \tag{3.10}$$

En changeant la fonction d'erreur, leur modèle n'essaie plus de minimiser la différence entre les étiquettes et la prédiction ($\epsilon = \tilde{m}_i - \hat{m}_i$) mais la différence entre la probabilité que la classe réelle du pixel soit 1 et la prédiction (sur une entrée s, $\epsilon = p(m_i = 1 | \tilde{m}_i, s) - \hat{m}_i$). Pour cela, la prédiction $\hat{m}_i$ est remplacée par une version adaptée au bruit : $(1 - \theta_1) * \hat{m}_i + \theta_0 * (1 - \hat{m}_i)$. Cette transformation est utilisée en Section 3.5.2.

Pour estimer les paramètres θ_0 et θ_1, on peut se baser sur une connaissance à priori. Dans le cas d'une simulation par exemple, le taux de bruit est connu. Dans une application réelle, le taux de bruit est inconnu, mais peut être appris. Sukhbaatar et Fergus [2014] proposent la méthode suivante. Après avoir entraîné leur réseau de neurones sur les données bruitées, ils ajoutent une couche sans activation et continuent l'entraînement. Ils montrent que la régularisation permet de s'assurer que les paramètres de cette dernière couche correspondent à la distribution du bruit. Entraîné sur des données bruitées, ils peuvent supprimer la dernière couche et obtenir des prédictions cohérentes avec des étiquettes non bruitées. Jindal *et al.* [2016] proposent une méthode équivalente, mais utilisent une couche de softmax régularisée à l'aide de dropout [Srivastava *et al.*, 2014]. Ils obtiennent de meilleurs résultats et la procédure est moins contraignante. En effet l'apprentissage ne se fait pas en

deux temps, et contrairement à Sukhbaatar et Fergus [2014], les poids peuvent être initialisés aléatoirement.

Vicente *et al.* [2016] veulent nettoyer des étiquettes de segmentation contenant des faux négatifs. Dans leur cas ce sont les ombres, pas les bâtiments, qui sont segmentées. Ils utilisent pour ça une technique qu'ils appellent *lazy annotations*, qui consiste à tracer un trait sur les deux ou trois ombres que l'utilisateur juge importantes, et un autre type de trait sur deux ou trois parties de l'image n'étant pas des ombres. Ces traits servent à attribuer une classe aux superpixels [Achanta *et al.*, 2012] de l'image et obtenir ainsi un masque grossier. Ils utilisent un *Least-Squares Support Vector Machines* [Suykens et Vandewalle, 1999] dont ils modifient la fonction de coût afin d'autoriser une proportion définie d'étiquettes à être modifiées de négatif vers positif, et une autre proportion de modifications inverses dans le but d'apprendre une segmentation de l'image à partir des étiquettes grossières créées précédemment.

Kaiser *et al.* [2017] essaient de savoir si on peut entraîner un modèle sur des étiquettes provenant d'OSM et obtenir une segmentation de qualité. Celle-ci est mesurée à partir d'étiquettes propres, décrivant parfaitement la segmentation attendue. Ces étiquettes sont chères à produire et ne peuvent donc pas être obtenue pour tout le jeu de données. Leurs évaluations concernent la segmentation (et non pas la détection de contours) de bâtiments et de routes. Ils répondent à cette problématique en pré entraînant un FCN pour la segmentation sémantique [Long *et al.*, 2015a] sur des données OSM et des images de Google Maps de Chicago, Paris, Zurich, et Berlin, pour l'utiliser ensuite sur 3 images de ISPRS et des étiquettes propres à Postdam. Leurs expérimentations mettent en évidence le fait qu'un ensemble d'apprentissage bruité peut être compensé par sa taille. À ensemble d'apprentissage de taille constante, utiliser les étiquettes d'OSM donne des performances 10% moins bonnes par rapport à l'utilisation d'étiquettes propres. Cependant, l'utilisation d'un plus grand ensemble d'apprentissage bruité mène à un meilleur score que l'apprentissage sur le petit jeu de données correctement étiqueté. L'accès à un ensemble d'apprentissage plus grand est rendu possible par la disponibilité des étiquettes OSM à grande échelle.

3.5.2 Proposition de modèles renforcés

Un modèle renforcé est un modèle qui a été conçu spécifiquement pour résister au bruitage des étiquettes. On compare le modèle de fusion présenté en Section 3.4 avec trois modèles renforcés par la modification de la prédiction avant le calcul du coût. Le modèle «Mnih» est extrait de la littérature Mnih et Hinton [2012], alors que les modèles «log-loss» et «de Bayes» sont des propositions.

Les transformations partent d'un principe commun : étant donné la nature du bruit étudié (suppression aléatoire d'objets à détecter), on va autoriser le modèle renforcé à créer des bâtiments qui n'apparaissent pas dans la vérité terrain. À cette fin, la fonction de coût est modifiée de manière à diminuer la pénalité subie par le modèle lorsqu'il classifie un pixel dans la classe bâtiment alors que la vérité terrain indique une classe non-bâtiment.

On rappelle la notation de la vérité terrain ne comportant aucun bruit y et de ses pixels m_i, de l'étiquette bruitée utilisée $\tilde{y}$ et de ses pixels $\tilde{m}_i$, ainsi que de la

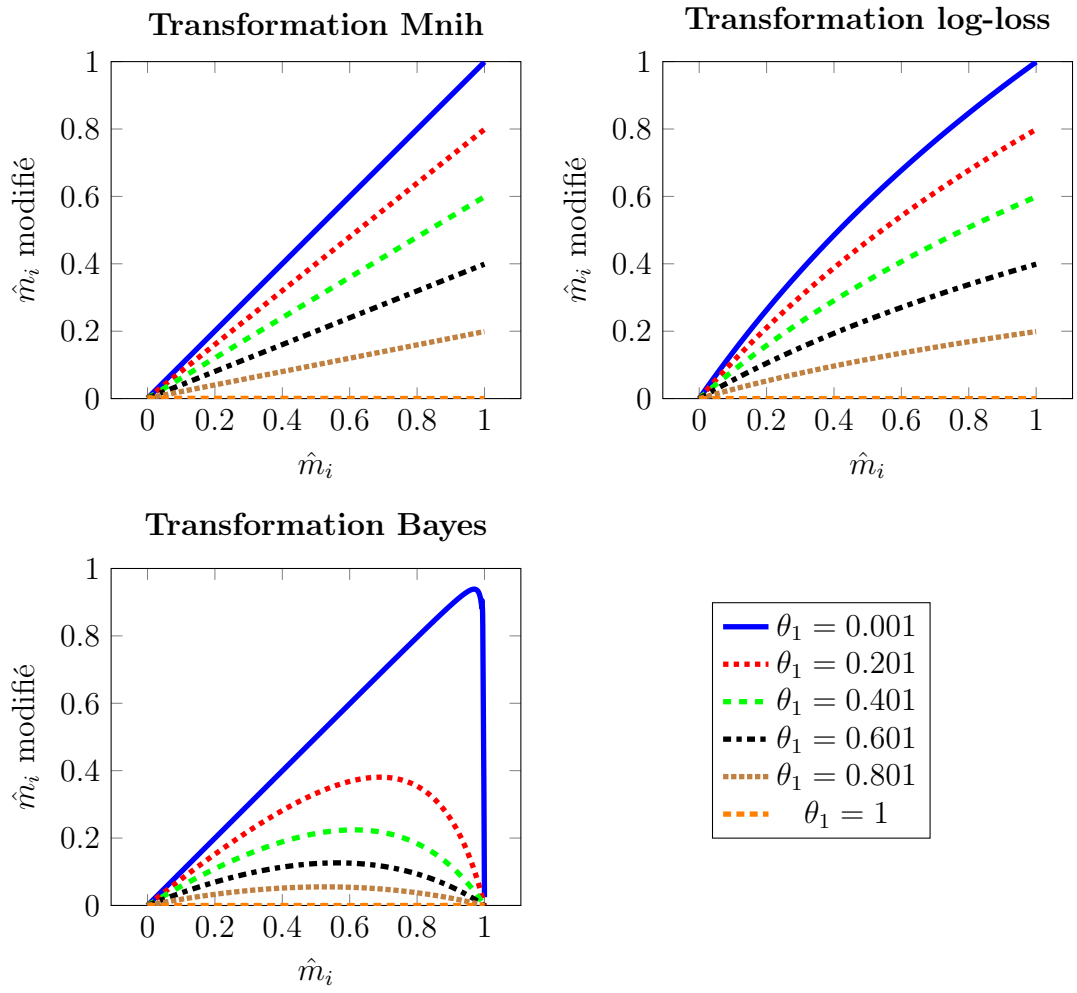

FIGURE 3.26 – Modification de $\hat{m}_i$ en fonction du taux de faux négatifs lorsque l'étiquette potentiellement bruitée est 0. θ_0 est fixé à 0.001. Plus il y a de faux négatifs, plus il y a de chance qu'une prédiction à 1 soit comparée à une fausse étiquette à 0. Les prédictions sont toutes réduites. Pour diminuer l'erreur.

prédiction du modèle $\hat{y}$ et de ses pixels $\hat{m}_i$.

On rappelle également que la définition des taux de faux positifs θ_0 et faux négatifs θ_1 au sein des étiquettes est défini en équation 3.10.

La première transformation est celle proposée par Mnih et Hinton [2012], étudiée en Section 3.5.1. Avant de calculer l'entropie croisée de la prédiction $\hat{y}$ et de l'étiquette bruitée $\tilde{y}$, ils appliquent la transformation suivantes aux éléments $\hat{m}_i$ de $\hat{y}$:

$$\hat{m}_i := \theta_0 * (1 - \hat{m}_i) + (1 - \theta_1) * \hat{m}_i \tag{3.11}$$

On conserve la fonction d'erreur de Jaccard et on applique la même transformation aux prédictions. Ce modèle est appelé modèle «Mnih» par la suite. La fonction est tracée en Figure 3.26 pour plusieurs valeurs de θ_1. On voit que plus le taux de

faux négatifs est élevé, plus on va minimiser la prédiction. Le modèle peut alors faire des prédictions élevés (il prédit avec une confiance élevée qu'un bâtiment est présent à cet endroit) et la transformation s'assure de diminuer cette valeur afin que la fonction d'erreur la pénalise moins.

La transformation de Mnih est linéaire, elle n'est donc pas impactée par l'assurance du modèle dans sa prédiction. Nous souhaitons évaluer des transformations pénalisant une prédiction fausse avec une moins grande intensité lorsque le modèle est sûr de lui. À ces fins nous proposons les deux transformations suivantes.

3.5.2.1 Transformation log-loss

Nous proposons le modèle que l'on appelle «log-loss», appliquant la transformation suivante :

$$\begin{aligned} \hat{m}_i := \tilde{m}_i * \frac{1 - (1 - \theta_0) * log((1 - \hat{m}_i) + 1)}{log(2)} \\ + (1 - \tilde{m}_i) * \frac{(1 - \theta_1) * log(\hat{m}_i + 1)}{log(2)} \end{aligned} \tag{3.12}$$

Puisque $\hat{m}_i$ est compris entre 0 et 1, $log(\hat{m}_i + 1)$ comme $log((1 - \hat{m}_i) + 1)$ sont compris entre 0 et $log(2)$. La division par $log(2)$ permet de ramener ce quotient entre 0 et 1. la multiplication par $1 - \theta$ permet ensuite comme pour Mnih d'obtenir une courbe croissante de 0 à $1 - \theta$.

Elle permet d'aplatir la courbe d'erreur quand le modèle donne une réponse fausse avec beaucoup d'assurance. Plus le modèle est sûr de lui, plus on lui donne le bénéfice du doute et donc plus la diminution de sa prédiction va être forte. Ainsi si l'étiquette vaut 0 (le pixel n'est pas étiqueté comme un pixel de bâtiment), on ne pénalise pas beaucoup plus le modèle s'il répond 1 que s'il répond 0.9. Plus le taux de faux négatif est grand, plus la différence de pénalité entre ces deux réponses est petite. Cette transformation est illustrée en Figure 3.26.

3.5.2.2 Transformation de Bayes

La troisième transformation que nous proposons pousse cette idée encore plus loin. à partir d'un certain seuil, on va faire plus confiance au modèle qu'aux étiquettes. À tel point que l'erreur va décroître si le modèle est assez sûr de lui. La transformation appliquée est la suivante :

$$\begin{aligned} \hat{m}_i := \tilde{m}_i * (1 - \frac{(1 - \theta_1) * \hat{m}_i}{(1 - \theta_1) * \hat{m}_i + \theta_0 * (1 - \hat{m}_i)} - \hat{m}_i) \\ - (1 - \tilde{m}_i) * (\frac{\theta_1 * \hat{m}_i}{\theta_1 * \hat{m}_i + (1 - \theta_0) * (1 - \hat{m}_i)} + \hat{m}_i) \end{aligned} \tag{3.13}$$

On désignera ce modèle comme le «modèle de Bayes». En effet, l'équation précédente est obtenue grâce à la loi de Bayes :

D'après celle ci,

$$p(m_i = 1|\tilde{m}_i) - \hat{m}_i = \frac{p(\tilde{m}_i|m_i = 1) * p(m_i = 1)}{p(\tilde{m}_i)} - \hat{m}_i \tag{3.14}$$

et puisque m_i ne peut valoir que 0 ou 1,

$$p(\tilde{m}_i) = p(\tilde{m}_i|m_i = 1) * p(m_i = 1) + p(\tilde{m}_i|m_i = 0) * p(m_i = 0) \tag{3.15}$$

Les définitions des paramètres θ_0 et θ_1 nous permettent d'obtenir via la loi de Bernoulli les deux distributions suivantes :

$$\begin{aligned} p(\tilde{m}_i|m_i = 0) &= \theta_0^{\tilde{m}_i} * (1 - \theta_0)^{(1-\tilde{m}_i)} \\ p(\tilde{m}_i|m_i = 1) &= \theta_1^{(1-\tilde{m}_i)} * (1 - \theta_1)^{\tilde{m}_i} \end{aligned} \tag{3.16}$$

puisque $p(m_i = 1) = 1 - p(m_i = 0)$ et $p(m_i = 1) = \hat{m}_i$ on obtient si $\tilde{m}_i = 0$

$$p(m_i = 1|\tilde{m}_i) - \hat{m}_i = \frac{\theta_1 * \hat{m}_i}{\theta_1 * \hat{m}_i + (1 - \theta_0) * (1 - \hat{m}_i))} - \hat{m}_i \tag{3.17}$$

et si $\tilde{m}_i = 1$

$$p(m_i = 1|\tilde{m}_i) - \hat{m}_i = \frac{(1 - \theta_1) * \hat{m}_i}{(1 - \theta_1) * \hat{m}_i + \theta_0 * (1 - \hat{m}_i)} - \hat{m}_i \tag{3.18}$$

La transformation obtenue est tracée en Figure 3.26.

3.5.3 Protocole expérimental

Pour mesurer l'impact d'une vérité terrain bruitée sur la qualité de l'apprentissage, nous utilisons une vérité terrain non bruitée, que nous bruitons de manière contrôlée. Nous notons que cette vérité terrain contient toutefois des erreurs, mais comme elles ne sont ni quantifiables ni corrigibles en temps et coût raisonnables, nous la considérons comme virtuellement non bruitée. Après avoir entraîné notre modèle sur une vérité terrain bruitée, nous l'évaluons sur une vérité terrain non bruitée. L'opération est répétée pour plusieurs niveaux de bruits afin d'évaluer la résistance du modèle.

3.5.3.1 Création d'un jeu de données bruité

Le contrôle de la détérioration permet de la quantifier, afin d'en évaluer l'impact sur la qualité de l'apprentissage. On cherche à avoir le moins de paramètres possibles à faire varier, l'un d'eux étant le taux de bruit introduit. Dans le cas de la suppression, le taux de bruit est le seul paramètre, correspondant à la probabilité pour chaque bâtiment d'être supprimé ou non. Les autres transformations nécessitent un paramétrage plus complexe. La translation par exemple nécessitera en plus de la sélection des bâtiments à translater de décider du vecteur de translation, avec une gestion

de la superposition de plusieurs bâtiments. Pour la fusion, il faut d'abord sélectionner les candidats à une possible fusion, ce qui n'est pas trivial. Il existe un risque pour que ces candidats ne soient pas assez nombreux pour impacter significativement l'apprentissage, diminuant l'intérêt de l'étude. Il en va de même pour la séparation. La simplification est facilement implémentable et peut être paramétrer de la même manière que la suppression. Cependant, beaucoup de bâtiments sont déjà convexes et le même risque que pour la fusion ou la séparation existe. L'ajout de bâtiment est le bruit le plus complexe à paramétrer. Il faut décider où ajouter un bâtiment et quelle forme lui donner. Il faut définir également à quoi correspond un taux de bruit de 100%. On voit donc que la suppression est une transformation permettant un protocole expérimental plus simple, nécessitant moins d'expérimentations que les alternatives. Cela augmente la possibilité de reproduire ce même protocole pour évaluer d'autre modèles et les comparer aux modèles proposés ici.

Pour plusieurs taux de bruit donnés, on entraîne le modèle sur les données bruitées obtenues, puis on évalue la qualité des prédictions sur un jeu de données non bruité. Tracer les courbes d'évaluation des performances en fonction du taux de bruit permet d'évaluer la résistance au bruit des modèles entraînés.

Parmi toutes les transformations possibles, nous avons choisi de nous concentrer sur la suppression de bâtiments. En effet, elle correspond à un cas commun au sein des données OSM : les zones non cartographiées. En étudiant la résistance à la suppression on peut connaître les performances d'un modèle quand tous les bâtiments n'ont pas été étiqueté, et ainsi juger du nombre de bâtiments qu'il sera nécessaire d'étiqueter à la main avant de pouvoir automatiser le processus. La reproductibilité de notre étude a aussi guidé notre choix. En effet, la suppression fait intervenir moins de paramètres et de calculs que les autres transformations.

3.5.3.2 Apprentissage

On entraîne le modèle de fusion décrits en section 3.4 sur un jeu de données de plus en plus bruité. On fixe d'abord la probabilité qu'un bâtiment soit supprimé, et on tire au sort la suppression ou non de chaque bâtiment présent dans les données étiquetées. Une fois les bâtiments supprimés, on fabrique les nouveaux masques correspondant à la vérité terrain bruitée et on entraîne les modèles voulus à segmenter les images satellites à partir de ces nouveaux masques.

On renforce ensuite ce modèle en modifiant sa prédiction avant le calcul de l'erreur. On compare ainsi la résistance au bruit du modèle existant, d'un modèle extrait de la littérature, et de deux propositions. Ces modèles sont décrits en Section 3.5.2.

Le jeu de données de la ville de Vegas est utilisé, séparé en trois parties : le jeu de données d'apprentissage bruité, le jeu de données de validation bruité de façon similaire au jeu d'apprentissage, et le jeu de données de test non bruité. Le jeu de validation sert à sélectionner l'époque d'apprentissage donnant le meilleur score. Ce sont les poids de cette époque qui sont utilisés pour obtenir le score final du modèle calculé sur le jeu de test non bruité.

Les modèles U-Net, décrits en Section 3.4 sont entraînés avec des masques obtenus à partir de données bruitées. On évalue le F1-score de ces modèles pour chaque époque sur un jeu de données bruité de validation (non utilisé pour l'optimisation

des paramètres) et on sélectionne l'époque ayant obtenu le meilleur F1-score. Les prédictions de ces modèles sont utilisées en entrée du modèle de fusion (voir Section 3.4.2.3) qui est entraîné avec les mêmes étiquettes bruitées, et dont la meilleure époque est choisie avec le même jeu de validation bruité.

Si le modèle sur-apprend (phénomène expliqué en Section 2.2.1), il apprendra à reproduire le bruit présent dans les données bruitées et obtiendra donc un score faible quand il sera comparé à des étiquettes non bruitées. Utiliser des étiquettes propres pour la validation permettrait de contourner ce problème. L'époque sélectionnée serait la meilleure époque face à des étiquettes non bruitées, avant que le modèle ne se soit sur-spécialisé avec des données bruitées. Cependant, cette solution n'est pas applicable en situation réelle. Si l'on voulait appliquer nos modèles sur les données d'OSM pour les nettoyer, nous n'aurions pas à notre disposition de données propres durant l'apprentissage pour obtenir un score de validation. Il est préférable de laisser le modèle sur apprendre plutôt que placer notre étude dans un contexte différent de l'application à OSM.

Le modèle de fusion sélectionné durant l'apprentissage est évalué sur des données non bruitées. Le score obtenu montre dans quelle mesure un modèle entraîné sur des données bruitées peut produire des bâtiments proches d'une vérité terrain virtuellement non bruitée. Cela permet d'estimer à quel point un modèle entraîné sur les données d'OSM (bruitées) peut produire des bâtiments proches de ceux étant effectivement présents dans la réalité.

Si le F1-score obtenu par les prédictions du modèles par rapport aux étiquettes non bruitées est meilleur que le F1-score obtenu par les étiquettes bruitées par rapport aux étiquettes non bruitées, cela signifie que le modèle peut produire des données de meilleure qualité que celles avec lesquelles il a été entraîné, et donc améliorer ces dernières.

L'introduction de bruit dans les données étant aléatoire, il est possible que pour un même taux de bruit, un modèle n'obtienne pas les même performances. On évalue donc le modèle trois fois pour chaque taux de bruit. Cette triple évaluation a été réalisée sur le modèle non renforcé. Elle a révélé des marges d'erreur très faibles, voir négligeables pour des taux de bruit plus élevés. Motivé par cette faible variance et par la constatation du temps de calcul prohibitif (plusieurs jours par évaluation sur une machine disposant de 100Go de RAM et 2 GPUs Nvidia K80), nous n'avons procédé qu'à une seule évaluation pour les autre modèles.

3.5.4 Résultats

3.5.4.1 Performances

La Figure 3.27 montre les courbes d'évaluation des modèle étudiés, ainsi que du modèle théorique «modèle étiquettes bruitées» (MEB) trouvant exactement une proportion p des bâtiments de la vérité terrain.

Ce dernier score est obtenu analytiquement. Si on appelle p la probabilité qu'un bâtiment soit supprimé, alors $precision(MEB; p) = 1$. En effet la probabilité qu'un bâtiment présent dans les étiquettes bruitées soit présent dans les étiquettes non bruitées est 1 car aucun bâtiment n'a été ajouté. Aussi $rappel(MEB; p) = 1 - p$. En

effet, la probabilité qu'un bâtiment présent dans les données non bruitées soit aussi présent dans les donnés bruitées est équivalente à la probabilité qu'il n'ait pas été supprimé. Le F1-score, défini par $F_1(model; p) = 2 \cdot \frac{\text{precision} \cdot \text{rappel}}{\text{precision} + \text{rappel}}$ est aura donc pour valeur $F_1(MEB; p) = 2 - \frac{2}{2-p}$.

On voit que le modèle de fusion, sans transformation de sa prédiction, présente déjà une certaine résistance au bruit. Lorsque plus de 30% des bâtiments sont supprimés il produit des étiquettes de meilleure qualité qu'un modèle théorique dont les prédictions seraient autant affectées par le bruit que les données d'apprentissage. Ça signifie que le modèle peut déjà être utilisé pour détecter des bâtiments manquants au sein des étiquettes.

Les courbes de rappel permettent de savoir quelle proportion de bâtiments réellement présents ont été détectés. On voit que les modèles log-loss et Mnih sont capables de détecter 80% des bâtiments quand ils n'ont été entraînés qu'avec 40% des bâtiments étiquetés. C'est encore vrai quand il sont entraîné avec seulement 20% des bâtiments étiquetés ($p = 0.8$). Pour $p = 0.9$, le modèle log-loss est largement plus performant que le modèle Mnih. Le premier retrouve plus de 20% des bâtiments quand le second n'en retrouve que 8%, soit moins que les 10% attendus. Nous avançons que les performances pour les niveaux de bruit élevés sont les plus importantes. En effet, lorsque des étiquettes sont difficiles ou chères à produire manuellement, il est intéressant de n'avoir que peu d'objet à étiqueter avant qu'un modèle ne puisse le faire automatiquement. Cette différence de performance entre l'état de l'art et notre proposition est donc significative.

Les courbes de précision permettent d'évaluer la quantité de bruit ajoutée par les modèles. Ce sont des bâtiments que le modèle a détectés alors qu'ils ne sont pas présents dans les étiquettes propres. Le bruit ajouté par le modèle ne correspond pas à des suppressions mais à des ajouts de bâtiments. Il est possible que certains d'entre eux soient réellement présents mais non étiquetés. On voit qu'à partir de $p = 0.6$ le modèle log-loss commence à voir sa précision décroître. En l'autorisant à créer de nouveaux bâtiments, on prend le risque d'en créer alors qu'il ne le faudrait pas.

Il est nécessaire de noter qu'un bâtiment identifié mais dont le contour est trop imprécis (ou fusionné avec un voisin) ne sera pas associé à l'étiquette correspondante, faisant à la fois baisser le rappel et la précision. Ces cas expliquent les valeurs initiales de rappel et de précision, pour $p = 0$.

Les expérimentations de Mnih et Hinton [2012] ont été faites pour un taux de faux négatifs à 5%, ce qui est un taux négligeable quand on voit ici que la résistance au bruit ne se fait sentir qu'après un taux de faux négatifs à 40%. Nos expérimentations permettent de vérifier que leur proposition permet une résistance au bruit même pour des valeurs de p plus élevées, puisqu'avec 40% des bâtiments étiquetés, le F1-score du modèle n'est presque pas dégradé. Cependant, contrairement à notre proposition, nous n'avons pas été capable d'entraîner le modèle pour des proportions de bâtiments supprimés plus élevées.

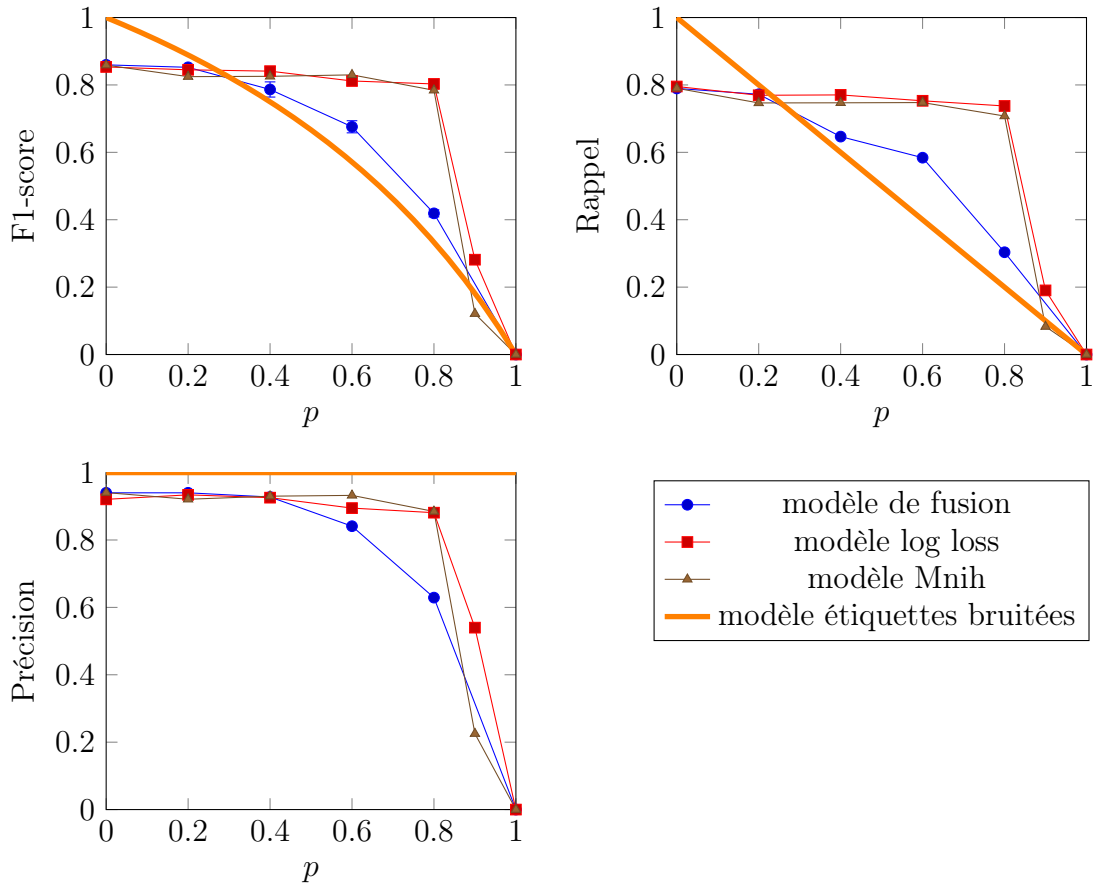

FIGURE 3.27 – Les modèles étudiés sont comparés à un modèle théorique dont la sortie est bruitée de la même manière que les étiquettes. Le modèle de Bayes ne figure pas dans ce graphique car nous ne sommes pas parvenus à l'entraîner.

3.5.4.2 Limites de l'approche

Au fur et à mesure de la dégradation des données, il est de plus en plus difficile d'entraîner les modèles. Les classes bâtiment et non bâtiment sont de plus en plus déséquilibrées, de moins en moins d'étiquettes bâtiments sont disponibles et de plus en plus d'images de bâtiments étiquetés comme non bâtiment viennent perturber le modèle. Pour ces raisons, le taux d'apprentissage doit être ajusté. Utiliser un taux d'apprentissage trop faible augmente excessivement le temps d'apprentissage et nous oblige à augmenter le nombre d'époques pour obtenir les mêmes performances, ce qui n'est pas possible dans des conditions où le temps est limite. C'est la raison pour laquelle il est nécessaire d'utiliser le taux d'apprentissage le plus élevé possible. Alors qu'un taux d'apprentissage de 10^{-4} peut être utilisé pour des taux de bruit faibles (< 0.2), il a été nécessaire d'utiliser un taux de 10^{-7} pour les taux de bruits les plus élevés. Une modification cyclique du taux d'apprentissage [Smith, 2017] nous permet de ne pas avoir à chercher le taux optimal pour chaque modèle. On définit les bornes

de recherche, ici 10^{-4} et 10^{-7}, et le taux d'apprentissage sera mis à jour à la fin de chaque époque de manière à sinuer entre ces deux valeurs.

Il peut être long d'entraîner et évaluer un modèle. Pour obtenir le F1-score de notre modèle étant donné une probabilité de suppression de bâtiment fixée, il faut compter entre trois et quatre jours de calculs sur une machine disposant d'une carte graphique Tesla K40 avec 11Go de mémoire dédiée et 110Go de Mémoire vive. La durée de cet entraînement limite le nombre possible d'expérimentations. On notera cependant que ces apprentissages sont indépendants et peuvent donc être réalisés en parallèle, à condition de disposer du matériel nécessaire.

Les résultats obtenus ne sont valables que pour la suppression de bâtiments. La courbe du F1-score des étiquettes bruitées serait différente pour une translation par exemple. Si l'amplitude de la translation est telle que chaque bâtiment translaté ne corresponde plus au bâtiment d'origine, la précision comme le rappel auront une valeur de $1-p$, amenant à un F1-score d'équation $F_1 = 1-p$. Chaque type de bruit devrait être étudié séparément.

Alors que le modèle de fusion a pu être évalué trois fois pour chaque niveau de bruit, les deux autres modèles n'ont pu l'être qu'une seule fois, par manque de temps. Le peu de variance dans les résultats du modèle de fusion laissent néanmoins penser que cette limite est mineure.

Nous n'avons pas été capables d'entraîner le modèle de Bayes, parce qu'en partant de poids initiaux aléatoires le modèle tombe dès la première époque dans un état où il ne prédit que des valeurs nulles, atteignant une perte nulle. En procédant comme Sukhbaatar et Fergus [2014] à un pré apprentissage du modèle sans modification, puis un ré-apprentissage à partir de cet état en ajoutant la modification voulue, nous avons espoir de pouvoir atteindre un état où le modèle apprend à résister au bruit sans sur-apprendre et annuler tous ses poids.

Enfin, on notera qu'un taux de bruit non négligeable dans les données considérées virtuellement propres pourrait fausser les résultats de cette étude. Cependant, le nombre de faux négatifs est assez faible pour qu'une vérification manuelle des données n'en dévoile aucun.

3.5.5 Conclusion

Nous avons vu qu'à partir d'un étiquetage de 20% des bâtiments nous pouvions retrouver 80% des bâtiments. On peut alors se demander s'il est possible d'utiliser les étiquettes produites pour entraîner à nouveau le modèle, en construisant l'ensemble d'entraînement à partir des prédictions. On se rapproche ici des techniques dites d'apprentissage faiblement supervisé, où une partie des étiquettes est proprement étiquetée, et une autre partie est étiquetée suivant une distribution de probabilité bruitée, fournie ici par le premier modèle.

Il est possible que les bâtiments identifiés par le modèle robuste aient des caractéristiques communes, telles que leur taille, la couleur de leur toit, leur forme, leur emplacement dans l'image, etc. Cela sortirait les nouvelles étiquettes obtenues de notre contexte d'expérimentation. En effet, le bruitage des données ne correspond alors plus à une suppression des bâtiments de manière aléatoire mais est biaisée par ces points communs.

De plus, les erreurs introduites par le modèle ne sont pas forcément des suppressions. Il peut produire des faux positifs (détection d'un bâtiment inexistant), fusionner des bâtiments trop proches les uns des autres, etc. Aussi, pour un bâtiment détecté on sait que le polygone produit a un IoU supérieur à 0.5 avec le polygone de la vérité terrain, rien ne garanti que le polygone décrit mieux le bâtiment. Cette déformation est encore une autre forme de bruit dont on ne connaît pas les répercussions sur l'apprentissage.

Il serait donc intéressant d'étudier les performances d'un modèle entraîné sur des prédictions afin de savoir à quel point les étiquettes peuvent être améliorées dans ce procédé.

Nous avons proposé un protocole expérimental permettant d'évaluer la résistance au bruit d'un modèle de segmentation lorsque toutes les instances de la classe segmentée ne sont pas étiquetées. Nous avons utilisé ce protocole afin d'évaluer un modèle de référence, n'ayant subit aucune adaptation au problème, une adaptation au problème tirée de l'état de l'art, et deux adaptations que nous proposons. L'adaptation de l'état de l'art s'est révélée inefficace pour des taux de bruits élevés car nous n'avons pas réussi à l'entraîner. Une de nos proposition, la transformation «de Bayes» n'a pas du tout pu être entraînée. L'autre proposition, la transformation «log-loss» a permis d'atteindre une résistance marquée au bruit, obtenant un score équivalent à la détection de 50% des bâtiments quand le modèle n'a été entraîné qu'avec 20% des bâtiments étiquetés.

On s'attachera dans une future étude à évaluer la résistance à différents bruits. On estimera également l'amélioration maximale pouvant être apportée aux étiquettes par un modèle résistant au bruit entraîné sur ses propres prédictions.

Chapitre 4

Conclusion

En conclusion de ce manuscrit, nous récapitulons les problématiques de Qucit et comment nos travaux permettent d'y répondre et nous explicitons les contributions apportées durant ces trois années. Nous abordons ensuite les perspectives d'évolutions des travaux, en listant notamment les points qui permettraient d'obtenir un modèle plus performant, la mise en production des modèles au sein d'un produit abouti, et l'application des techniques et méthodes acquises pendant cette thèse à d'autres problématiques.

4.1 Récapitulatif des problèmes abordés

L'apprentissage automatique fut un sujet d'étude récurrent durant cette thèse, qu'il soit non supervisé, avec la détection de vélos défectueux au sein de systèmes de vélos en libre service, ou supervisé avec la détection de bâtiments au sein d'images satellites. Cette section fait le bilan des problématiques auxquelles Qucit fait face puis des contributions d'apprentissage automatique de cette thèse permettant d'y répondre.

4.1.1 Les problématiques de Qucit

Qucit travaille à l'amélioration des villes afin de les rendre plus agréable à vivre, plus efficaces, et plus durables en utilisant l'intelligence artificielle, sous forme d'apprentissage automatique. L'objet de cette thèse était de corriger les données utilisées par Qucit afin d'obtenir des analyses plus justes permettant de fournir un meilleur service aux clients ainsi qu'à leurs usagers. Dans un premier temps, nous nous sommes intéressés à la détection de vélos défectueux au sein des données de trajets du système de vélo en libre service de New York. Le manque de données étiquetées nous a poussés dans un second temps à corriger les données topographiques utilisées pour la contextualisation. À l'aide d'images satellites nous extrayons le contour de chaque bâtiment représenté, dans le but de vérifier l'exactitude des données contenues au sein des données ouvertes d'OSM.

L'application BikePredict permet aux utilisateurs de vélos en libre service de savoir à l'avance s'ils trouveront un vélo ou une place dans la station de vélos en

libre service de leur choix. Malheureusement cette prédiction peut être faussée par la présence de vélos défectueux. Pour éviter d'orienter les usagers vers une station ne contenant pas de vélo utilisable, et pour alerter les gestionnaires du système de vélos en libre service au plus vite, nous proposons un modèle d'analyse de données ouvertes de trajets de vélos en libre service pour la détection de vélos défectueux.

Les données fournies par les clients de Qucit telles que les données de trajets de vélos représentent des anomalies, mais certains jeux de données contiennent eux même des anomalies. C'est le cas des données topographiques, utilisées pour la contextualisation des données métiers. Elles sont récupérées dans les données ouvertes des villes, ou sur OSM. Les données provenant d'OSM respectent toujours le même format et les types d'objets pouvant être rencontrés sont connus d'avance. Cependant, ces données sont collaboratives. Ce sont les nombreux utilisateurs qui créent chaque élément. Les zones non cartographiées, plus à jour, ou erronées sont nombreuses et nécessitent une vérification voir une complétion manuelle. Nous avons donc étudié le problème de détection de bâtiments au sein d'images satellites, dans le but de corriger les données en provenance d'OSM et ainsi améliorer les services proposés par Qucit.

4.1.2 Contributions

On compte trois contributions majeures au sein de cette thèse.

La première contribution est une étude de la détection de vélos défectueux au sein de systèmes de vélos en libre service. On extrait des propriétés de couples (vélo, fenêtre de temps) en se basant sur les trajets de vélos exposés au sein de données ouvertes. Ces propriétés sont partitionnées dans le but d'identifier des comportements anormaux. Cette contribution a fait l'objet de deux publications. La première à la conférence française RFIA [Delassus *et al.*, 2016b] et la seconde à la conférence internationale SSCI [Delassus *et al.*, 2016a].

La deuxième contribution est une étude sur la fusion de modèles d'apprentissage profond pour la segmentation de bâtiments. On compare dans cette étude plusieurs modèles de fusion. On constate que notre proposition, l'utilisation d'un modèle d'apprentissage profond plutôt que la moyenne non pondérée des solutions, mène à une amélioration significative des résultats. Cette étude a fait l'objet d'une soumission à la conférence internationale CIKM. Rejetée avec des avis encourageants, elle sera corrigée et soumise à nouveau à une autre conférence internationale. Cette étude a également donné lieu à une participation au workshop «building detection» de la conférence CVPR 2018. Présenté sous forme de concours similaire à celui du challenge SpaceNet, nous nous y sommes classés sixième. Cette participation a donné lieu à la publication d'un article cours [Delassus et Giot, 2018] dans CVPRw et la présentation d'un poster à la conférence CVPR 2018.

La dernière contribution est une étude sur la résistance au bruit. Nous avons proposé un protocole expérimental permettant d'évaluer la résistance au bruit d'un modèle de segmentation lorsque toutes les instances de la classe segmentée ne sont pas étiquetées. Nous avons utilisé ce protocole afin d'évaluer un modèle de référence, n'ayant subit aucune adaptation au problème, une adaptation au problème tirée de l'état de l'art, et deux adaptations que nous proposons. L'adaptation de l'état de l'art

s'est révélée inefficace pour des taux de bruits élevés car nous n'avons pas réussi à l'entraîner. Une de nos propositions, la transformation «de Bayes» n'a pas du tout pu être entraînée. L'autre proposition, la transformation «log-loss» a permis d'atteindre une résistance marquée au bruit, obtenant un score équivalent à la détection de 50% des bâtiments quand le modèle n'a été entraîné qu'avec 20% des bâtiments étiquetés. Cette contribution sera prochainement l'objet d'une soumission.

4.2 Perspectives

Nous avons montré dans cette thèse que la détection de bâtiments au sein d'images satellites est un problème mettant en jeu plusieurs aspects de l'apprentissage automatique. L'approfondissement des études menées, l'amélioration des performances des modèles proposés, la mise en production des solutions et l'ouverture à d'autres types de problèmes sont des défis ouvrant la porte à de nombreux cas de recherche.

4.2.1 Approfondissement des études

Les études déjà menées durant cette thèse peuvent être approfondies afin d'en tirer plus d'enseignement ou pour élargir leur spectre d'applications.

4.2.1.1 Détection des vélos défectueux

Les travaux futurs viseront à améliorer le modèle en utilisant une meilleure fonction de coût, en concevant un nouvel ensemble de vérité terrain grâce à l'utilisation de méthodes standard de classification à sorties agrégées, et enfin en extrayant des caractéristiques qui nous permettront de tirer pleinement parti de l'algorithme de clustering. Un plus large panel d'hyper paramètres pourra être évalué grâce à des méthodes moins naïves telles qu'une optimisation Bayésienne.

4.2.1.2 Compatibilité des étiquettes et des architectures

On a vu en Section 3.3 que la transformation des polygones de la vérité terrain en différents types d'images étiquette débouchait sur des solutions aux performances différentes, et qu'une architecture adaptée à la prédiction d'un type d'étiquette n'était pas forcément adaptée à la prédiction d'un autre type d'étiquette. Par exemple, nous n'avons pas pu utiliser l'architecture SegNet pour prédire la distance aux bordures de bâtiments.

Étudier les performances de différentes architectures associées à différents types d'étiquettes permettrait d'étudier quelles combinaisons sont plus adaptées afin de mieux comprendre les raisons pour lesquelles une architecture serait plus adaptée à certaines étiquettes, ou au contraire les éléments d'une architecture l'empêchant d'atteindre des performances acceptables sur certaines étiquettes.

4.2.1.3 Amélioration d'étiquettes bruitées

Notre étude, détaillée en Section 3.5, permet de quantifier la résistance d'un modèle à la suppression de bâtiments au sein des étiquettes. Une de nos contributions est la proposition d'une modification permettant à notre modèle de détecter au moins 50% des bâtiments présents dans les images lorsque seulement 20% des bâtiments ont été étiquetés.

Il serait intéressant d'étudier si à partir de ces nouvelles étiquettes, incluant plus de 50% des bâtiments, il est possible d'entraîner un modèle permettant de détecter encore plus de bâtiments.

Aussi, ces étiquettes prédites sont susceptibles de contenir d'autres types de bruits que la suppression de bâtiments. Le modèle peut produire des faux positifs, et aura tendance à simplifier la forme des polygones. L'étude d'autres formes de bruit est donc nécessaire, afin de trouver un moyen d'obtenir un jeu de données étiqueté au mieux en faisant intervenir au minimum le travail d'étiquetage manuel.

4.2.2 Modèle plus performant

Les modèles soumis aux concours de détection de bâtiments ont toujours été des modèles faits pour étudier un problème bien défini, comme la fusion de modèles et le type d'étiquettes adéquat. Nous n'avons jamais eu l'occasion de mettre en place le modèle le plus performant possible. D'autres éditions du challenge SpaceNet seront l'occasion pour nous de nous intéresser au perfectionnement du modèle proposé.

Nous avons vu qu'une fusion de plusieurs modèles permettait d'améliorer le score final. Cependant, la méthode utilisée pour la fusion n'est pas le seul élément influant ce score. La nature des modèles fusionnés importe aussi. Le but ici est de fusionner des modèles se complétant les uns les autres afin que chacun remplisse un rôle particulier. Ces modèles peuvent varier dans leur architecture, dans le type d'étiquette qu'ils prédisent, dans la nature des objets prédits (on pourra par exemple utiliser un modèle prédisant les routes, un autre prédisant les surfaces d'eau, ou encore séparer les bâtiments par taille ou par couleur grâce à du partitionnement et y dédier des modèles). De plus, incorporer l'assemblage des différentes parties d'image découpées au sein de l'architecture permettrait d'assembler les modèles fusionnés et le combinateur en un seul modèle. Nous pourrions ainsi procéder à un apprentissage de bout en bout.

On note une difficulté de l'architecture U-Net utilisée à détecter à la fois des bâtiments de grande taille et des bâtiments de petite taille. Outre la solution de fusionner des modèles dédiés à la prédiction des bâtiments de taille donnée, l'utilisation de convolutions dilatées serait d'une aide non négligeable.

Les étiquettes utilisées dans l'étude de la fusion ainsi que dans l'étude de la résistance au bruit sont des masques. Or, on sait qu'il est possible d'obtenir de meilleures performances en prédisant les trois classes bâtiment, fond, bordure de bâtiments, ou en prédisant la distance aux bordures.

La segmentation d'instance permet non plus de segmenter l'image en différentes classes, mais en différentes instances de ces classes. On pourrait ainsi régler le problème de bâtiments adjacents étant détectés comme un seul bâtiment. Le modèle

Mask-R CNN [He *et al.*, 2017] est un exemple d'architecture pouvant être utilisée à ces fins.

Finalement, le post-traitement est inexistant dans notre processus. L'utilisation de champs aléatoires conditionnels [Lafferty *et al.*, 2001] permet généralement d'affiner la prédiction, résultant en une séparation des bâtiments proches les uns des autres et une meilleure définition des contours.

4.2.3 Outil de détection d'erreurs OSM

L'idée à l'origine de cette thèse était de mettre au point un outil permettant d'améliorer les données d'OSM. Une fois les bâtiments extraits des images satellites, une étape cruciale est manquante, celle d'utiliser ces données pour apporter une correction.

Pour cette raison, le développement d'un prototype a débuté sous la forme d'un projet pédagogique proposé pour l'unité d'enseignement «Projet Fin d'Année» au sein de l'École Nationale Supérieure d'Électronique, Informatique, Télécommunications, Mathématique et Mécanique de Bordeaux. Sept élèves de deuxième année ont travaillé à la réalisation d'un outil graphique permettant de faciliter la correction et la complétion des données OSM par les utilisateurs.

Cet outil se présente sous la forme d'une interface web à travers laquelle les utilisateurs peuvent visualiser les données d'OSM superposées aux images satellites, augmentées par des informations tirées de la détection de bâtiments. Des marqueurs sont ajoutés, indiquant des emplacements où des bâtiments ont été détectés mais sont manquants dans les données OSM, des emplacements où des bâtiments sont présents dans les données OSM mais n'ont pas été détectés, et enfin des emplacements où des bâtiments sont présents au sein des deux jeux de données mais leurs formes sont incompatibles.

L'utilisateur peut cliquer sur les marqueurs et ajouter, supprimer, ou modifier les polygones pour corriger les données OSM. En s'identifiant avec son compte OSM, il peut envoyer les corrections pour qu'elles soient appliquées et disponibles à tous. Des captures d'écran de l'outil sont présentées en Figure 4.1.

Ce prototype montre que la mise en production d'un produit fini est désirable. Il faudrait pour cela que les modèles utilisés soient entraînés sur les données OSM, avec les contours de bâtiments et les images satellites que l'on retrouve sur le site d'OSM. Il reste également à régler les problèmes au frontières des images. En effet, les modèles sont entraînés et évalués image par image, sans jamais se soucier des bâtiments étant séparés sur plusieurs images. Les images satellites utilisées par OSM étant petites (256x256 pixels), ce cas de figure est abondant.

4.2.4 Application à d'autres problèmes

Les bâtiments ne sont pas les seuls éléments topographiques utilisés pour contextualiser les données. Les arbres et autres éléments de verdure, les routes, l'éclairage public sont autant d'éléments pouvant être extraits d'images satellites (de nuit pour l'éclairage) afin d'améliorer les données à notre disposition.

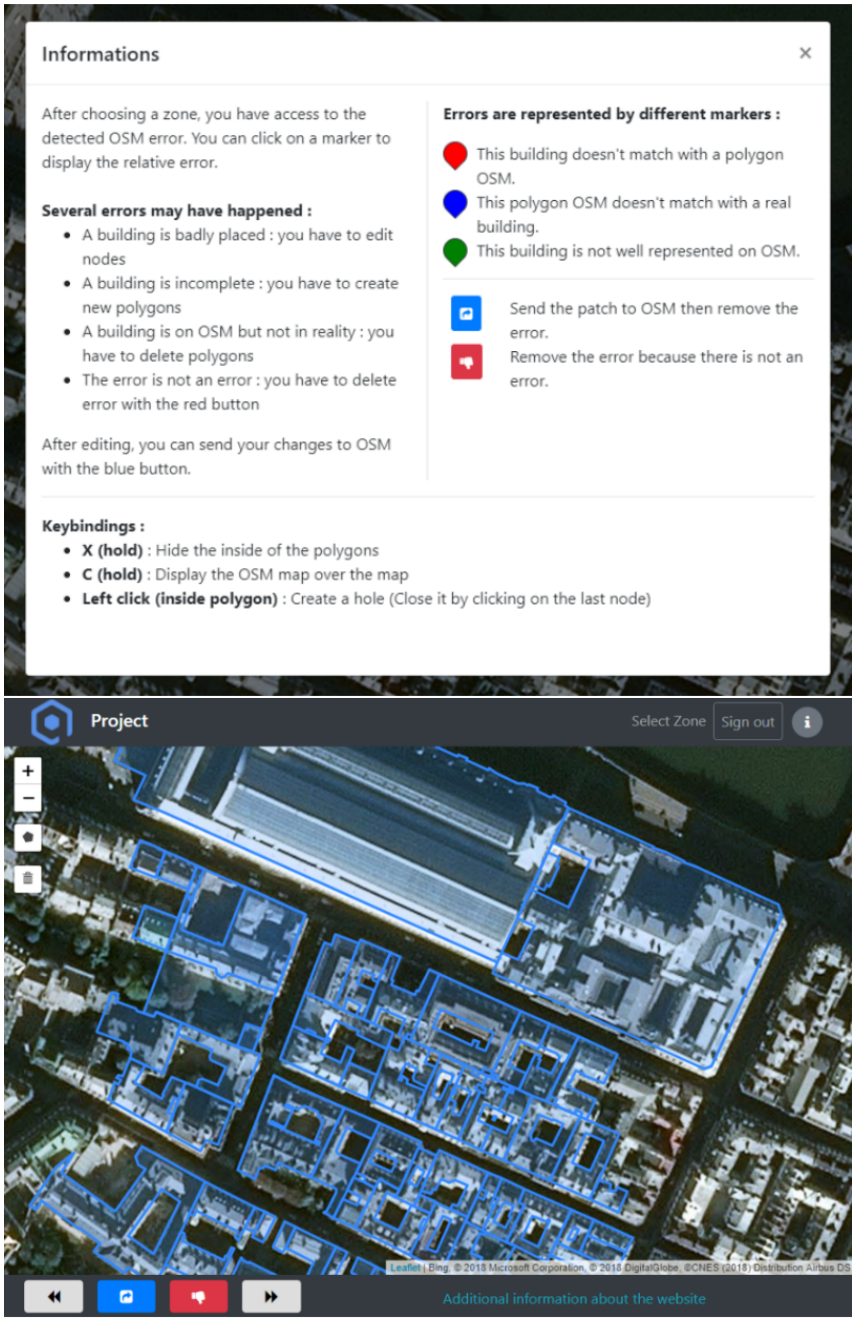

FIGURE 4.1 – Captures d'écran de l'outil développé par les élèves de l'ENSEIRB-MATMECA

Les images n'ont pas à être limitées aux images satellites pour la même application. Il est possible d'obtenir des images de plus haute résolution et mises à jour plus fréquemment en utilisant des ballons flottant au dessus des villes, comme le propose

Swiftera[1]. Dans ce cas là, il devient possible de surveiller le trafic automobile ou piéton pour développer à Qucit plus de solutions permettant d'améliorer la mobilité urbaine.

Enfin, la détection d'objets au sein d'images et la segmentation connaissent énormément d'applications, par exemple dans le domaine médicale. Une grande expertise est demandée aux médecins travaillant en imagerie médicale pour lire des résultats d'imagerie par résonance magnétique, scanners, radios... Une amélioration des outils existants dans ce domaine permettrait de réduire les coûts afin que plus de patients aient accès à ce type d'examens.

1. https://www.swiftera.co/

Bibliographie

Abadi, Martín, Agarwal, Ashish, Barham, Paul, Brevdo, Eugene, Chen, Zhifeng, Citro, Craig, Corrado, Greg S, Davis, Andy, Dean, Jeffrey, Devin, Matthieu *et al.*, 2016. Tensorflow : Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv :1603.04467*.

Abdallah, Aisha, Maarof, Mohd Aizaini et Zainal, Anazida, 2016. Fraud detection system : A survey. *Journal of Network and Computer Applications*, 68 :90–113.

Achanta, Radhakrishna, Shaji, Appu, Smith, Kevin, Lucchi, Aurelien, Fua, Pascal, Süsstrunk, Sabine *et al.*, 2012. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11) :2274–2282.

Aeschbach, Philipp, Zhang, Xiaojing, Georghiou, Angelos et Lygeros, John, 2015. Balancing bike sharing systems through customer cooperation–a case study on london's barclays cycle hire. Dans *IEEE Conference on Decision and Control*.

Ahmadi, Salman, Zoej, MJ Valadan, Ebadi, Hamid, Moghaddam, Hamid Abrishami et Mohammadzadeh, Ali, 2010. Automatic urban building boundary extraction from high resolution aerial images using an innovative model of active contours. *International Journal of Applied Earth Observation and Geoinformation*, 12(3) :150–157.

Ahmed, Mohiuddin, Mahmood, Abdun Naser et Islam, Md Rafiqul, 2016. A survey of anomaly detection techniques in financial domain. *Future Generation Computer Systems*, 55 :278–288.

Assaad, Mohammad, Boné, Romuald et Cardot, Hubert, 2008. A new boosting algorithm for improved time-series forecasting with recurrent neural networks. *Information Fusion*, 9(1) :41–55.

Badrinarayanan, Vijay, Kendall, Alex et Cipolla, Roberto, 2015. Segnet : A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv :1511.00561*.

Bai, Min et Urtasun, Raquel, 2017. Deep watershed transform for instance segmentation. Dans *CVPR, 2017*, pages 2858–2866. IEEE.

Bello, Irwan, Zoph, Barret, Vasudevan, Vijay et Le, Quoc V, 2017. Neural optimizer search with reinforcement learning. *arXiv preprint arXiv :1709.07417*.

Bertens, Roel, Vreeken, Jilles et Siebes, Arno, 2016. Beauty and brains : Detecting anomalous pattern co-occurrences. ArXiv :1512.07048.

Beucher, Serge et Meyer, Fernand, 1992. The morphological approach to segmentation : the watershed transformation. *Optical Engineering-New York-Marcel Dekker Incorporated-*, 34 :433–433.

Bin, Huang, Chen, Weihai, Wu, Xingming et Chun-Liang, Lin, 2017. High-quality face image SR using conditional generative adversarial networks. *CoRR*, abs/1707.00737.
URL http://arxiv.org/abs/1707.00737

Bishop, Christopher M, 2006. Pattern recognition. *Machine Learning*.

Bonnotte, Nicolas, Cherrier, Raphaël, Delassus, Rémi et Alouini, Yassine, 2015. Real-time data analytics and optimization of shared-vehicles networks. Dans *2nd ITS World Congres*.

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

Breiman, Leo, 2001. Random forests. *Machine Learning*, 45(1) :5–32. doi :10.1023/A:1010933404324.
URL https://doi.org/10.1023/A:1010933404324

Breunig, Markus M, Kriegel, Hans-Peter, Ng, Raymond T et Sander, Jörg, 2000. Lof : identifying density-based local outliers. Dans *ACM sigmod record*, tome 29, pages 93–104. ACM.

Brice, Claude R et Fennema, Claude L, 1970. Scene analysis using regions. *Artificial intelligence*, 1(3-4) :205–226.

Brochu, Eric, Cora, Vlad M et De Freitas, Nando, 2010. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv :1012.2599*.

Brun, Luc, 1996. *Segmentation d'images couleur a base Topologique*. Thèse de doctorat, Bordeaux 1.

Buczak, Anna L et Guven, Erhan, 2016. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, 18(2) :1153–1176.

Burda, Yuri, Edwards, Harri, Pathak, Deepak, Storkey, Amos, Darrell, Trevor et Efros, Alexei A, 2018. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv :1808.04355*.

Butun, Ismail, Morgera, Salvatore D et Sankar, Ravi, 2014. A survey of intrusion detection systems in wireless sensor networks. *IEEE communications surveys & tutorials*, 16(1) :266–282.

Canty, Morton J, 2014. *Image analysis, classification and change detection in remote sensing : with algorithms for ENVI/IDL and Python*. Crc Press.

Cauchy, Augustin, 1847. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847) :536–538.

Chandola, Varun, Banerjee, Arindam et Kumar, Vipin, 2009. Anomaly detection : A survey. *ACM Comput. Surv.*, 41(3) :15 :1–15 :58. doi :10.1145/1541880.1541882.
URL http://doi.acm.org/10.1145/1541880.1541882

Chapelle, Olivier, Vapnik, Vladimir, Bousquet, Olivier et Mukherjee, Sayan, 2002. Choosing multiple parameters for support vector machines. *Machine learning*, 46(1-3) :131–159.

Chaudhuri, D, Kushwaha, NK, Samal, A et Agarwal, RC, 2016. Automatic building detection from high-resolution satellite images based on morphology and internal gray variance. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 9(5) :1767–1779.

Chen, Chaoyue, Gong, Weiguo, Hu, Yan, Chen, Yongliang et Ding, Yi, 2017a. Learning oriented region-based convolutional neural networks for building detection in satellite remote sensing images. *ISPRS*, 42.

Chen, Liang-Chieh, Papandreou, George, Kokkinos, Iasonas, Murphy, Kevin et Yuille, Alan L, 2016. Deeplab : Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv :1606.00915*.

Chen, Liang-Chieh, Papandreou, George, Schroff, Florian et Adam, Hartwig, 2017b. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv :1706.05587*.

Chollet, François *et al.*, 2015. Keras. https://github.com/fchollet/keras.

Cinque, Luigi, Foresti, G et Lombardi, Luca, 2004. A clustering fuzzy approach for image segmentation. *Pattern Recognition*, 37(9) :1797–1807.

Ciresan, Dan, Giusti, Alessandro, Gambardella, Luca M et Schmidhuber, Jürgen, 2012. Deep neural networks segment neuronal membranes in electron microscopy images. Dans *Advances in neural information processing systems*, pages 2843–2851.

Cortes, Corinna et Vapnik, Vladimir, 1995. Support-vector networks. *Machine learning*, 20(3) :273–297.

Cover, Thomas et Hart, Peter, 1967. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1) :21–27.

Csáji, Balázs Csanád, 2001. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24 :48.

Cunningham, Pádraig, Cord, Matthieu et Delany, Sarah Jane, 2008. Supervised learning. *Machine learning techniques for multimedia*, pages 21–49.

De Brabandere, Bert, Neven, Davy et Van Gool, Luc, 2017. Semantic instance segmentation with a discriminative loss function. *arXiv preprint arXiv :1708.02551*.

DeKruger, David et Hunt, Bobby R, 1994. Image processing and neural networks for recognition of cartographic area features. *ICPR*, 27(4) :461–483.

Delassus, Remi et Giot, Romain, 2018. Cnns fusion for building detection in aerial images for the building detection challenge. Dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 242–246.

Delassus, Rémi, Giot, Romain, Cherrier, Raphael, Barbieri, Gabriele et Mélançony, Guy, 2016a. Broken bikes detection using citibike bikeshare system open data. Dans *Computational Intelligence (SSCI), 2016 IEEE Symposium Series on*, pages 1–7. IEEE.

Delassus, Rémi, Giot, Romain, Melançon, Guy et Cherrier, Raphael, 2016b. Détection de vélos inutilisables grâce aux données ouvertes du système de vélos en libre service citibike. Dans *Journée Transports Intelligents-RFIA 2016*.

Dhillon, Inderjit S, Mallela, Subramanyam et Kumar, Rahul, 2003. A divisive information-theoretic feature clustering algorithm for text classification. *Journal of machine learning research*, 3(Mar) :1265–1287.

Di Francescomarino, Chiara, Dumas, Marlon, Federici, Marco, Ghidini, Chiara, Maggi, Fabrizio Maria, Rizzi, Williams et Simonetto, Luca, 2018. Genetic algorithms for hyperparameter optimization in predictive business process monitoring. *Information Systems*, 74 :67–83.

Dias, Gabriel Martins, Bellalta, Boris et Oechsner, Simon, 2015. Predicting occupancy trends in barcelona's bicycle service stations using open data. Dans *SAI Intelligent Systems Conference (IntelliSys), 2015*, pages 439–445.

Dice, Lee R, 1945. Measures of the amount of ecologic association between species. *Ecology*, 26(3) :297–302.

Dolz, Jose, Desrosiers, Christian, Wang, Li, Yuan, Jing, Shen, Dinggang et Ayed, Ismail Ben, 2017. Deep cnn ensembles and suggestive annotations for infant brain mri segmentation. *arXiv preprint arXiv :1712.05319*.

Dumoulin, Vincent et Visin, Francesco, 2016. A guide to convolution arithmetic for deep learning. *ArXiv e-prints*.

Ertoz, Levent, Steinbach, Michael et Kumar, Vipin, 2002. A new shared nearest neighbor clustering algorithm and its applications. Dans *Workshop on clustering high dimensional data and its applications at 2nd SIAM international conference on data mining*, pages 105–115.

Ester, Martin, Kriegel, Hans-Peter, Sander, Jörg, Xu, Xiaowei *et al.*, 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. Dans *Kdd*, tome 96, pages 226–231.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J. et Zisserman, A., 2010. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2) :303–338.

Fanaee-T, Hadi et Gama, Joao, 2014. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, 2(2-3) :113–127.

Fishman, Elliot, 2016. Bikeshare : A review of recent literature. *Transport Reviews*, 36(1) :92–113.

Foo, Chuan-sheng, Do, Chuong B et Ng, Andrew Y, 2008. Efficient multiple hyperparameter learning for log-linear models. Dans *Advances in neural information processing systems*, pages 377–384.

Frénay, Benoît et Verleysen, Michel, 2014. Classification in the presence of label noise : a survey. *IEEE transactions on neural networks and learning systems*, 25(5) :845–869.

Freund, Yoav, 1995. Boosting a weak learning algorithm by majority. *Information and computation*, 121(2) :256–285.

Freund, Yoav, Schapire, Robert et Abe, Naoki, 1999. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780) :1612.

Freund, Yoav, Schapire, Robert E *et al.*, 1996. Experiments with a new boosting algorithm. Dans *Icml*, tome 96, pages 148–156. Citeseer.

Friedman, Jerome H, 2001. Greedy function approximation : a gradient boosting machine. *Annals of statistics*, pages 1189–1232.

Gast, Nicolas, Massonnet, Guillaume, Reijsbergen, Daniël et Tribastone, Mirco, 2015. Probabilistic forecasts of bike-sharing systems for journey planning. Dans *The 24th ACM International Conference on Information and Knowledge Management (CIKM 2015)*.

Ghaffarian, Salar, 2014. Automatic building detection based on supervised classification using high resolution google earth images. *ISPRS*, 40(3) :101.

GIOT, Romain et CHERRIER, RaphaÃ«l, 2014. Predicting bikeshare system usage up to one day ahead. Dans *IEEE Symposium Series in Computational Intelligence 2014 (SSCI 2014). Workshop on Computational Intelligence in Vehicles and Transportation Systems (CIVTS 2014)*, pages 1–8.

GIOT, Romain, HEMERY, Baptiste et ROSENBERGER, Christophe, 2010. Low cost and usable multimodal biometric system based on keystroke dynamics and 2d face recognition. Dans *ICPR, 2010*, pages 1128–1131. IEEE.

GIROLAMI, Mark et HE, Chao, 2003. Probability density estimation from optimally condensed data samples. *IEEE Transactions on pattern analysis and machine intelligence*, 25(10) :1253–1264.

GIRRES, Jean-François et TOUYA, Guillaume, 2010. Quality assessment of the french openstreetmap dataset. *Transactions in GIS*, 14(4) :435–459.

GIRSHICK, Ross, 2015. Fast r-cnn. Dans *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

GOMEZ-URIBE, Carlos A et HUNT, Neil, 2016. The netflix recommender system : Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)*, 6(4) :13.

GOODFELLOW, Ian, 2016. Nips 2016 tutorial : Generative adversarial networks. *arXiv preprint arXiv :1701.00160*.

GOODFELLOW, Ian, BENGIO, Yoshua, COURVILLE, Aaron et BENGIO, Yoshua, 2016. *Deep learning*, tome 1. MIT press Cambridge.

GOODFELLOW, Ian, POUGET-ABADIE, Jean, MIRZA, Mehdi, XU, Bing, WARDE-FARLEY, David, OZAIR, Sherjil, COURVILLE, Aaron et BENGIO, Yoshua, 2014. Generative adversarial nets. Dans *Advances in neural information processing systems*, pages 2672–2680.

GUO, Yandong, ZHANG, Lei, HU, Yuxiao, HE, Xiaodong et GAO, Jianfeng, 2016. Ms-celeb-1m : A dataset and benchmark for large-scale face recognition. Dans *European Conference on Computer Vision*, pages 87–102. Springer.

GUO, Yanming, LIU, Yu, GEORGIOU, Theodoros et LEW, Michael S, 2018. A review of semantic segmentation using deep neural networks. *International Journal of Multimedia Information Retrieval*, 7(2) :87–93.

HAKLAY, Mordechai, 2010. How good is volunteered geographical information ? a comparative study of openstreetmap and ordnance survey datasets. *Environment and planning B : Planning and design*, 37(4) :682–703.

HAKLAY, Mordechai et WEBER, Patrick, 2008. Openstreetmap : User-generated street maps. *IEEE Pervasive Computing*, 7(4) :12–18.

HAMAGUCHI, Ryuhei et HIKOSAKA, Shuhei, 2018. Building detection from satellite imagery using ensemble of size-specific detectors. Dans *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

HE, Kaiming, GKIOXARI, Georgia, DOLLÁR, Piotr et GIRSHICK, Ross, 2017. Mask r-cnn. Dans *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE.

HE, Kaiming, ZHANG, Xiangyu, REN, Shaoqing et SUN, Jian, 2014. Spatial pyramid pooling in deep convolutional networks for visual recognition. Dans *European Conference on Computer Vision*, pages 346–361. Springer.

HE, Kaiming, ZHANG, Xiangyu, REN, Shaoqing et SUN, Jian, 2015. Delving deep into rectifiers : Surpassing human-level performance on imagenet classification. Dans *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

HE, Kaiming, ZHANG, Xiangyu, REN, Shaoqing et SUN, Jian, 2016. Deep residual learning for image recognition. Dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

HE, Zengyou, XU, Xiaofei et DENG, Shengchun, 2003. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9-10) :1641–1650.

HEARST, Marti A., 1998. Support vector machines. *IEEE Intelligent Systems*, 13(4) :18–28. doi :10.1109/5254.708428.
URL http://dx.doi.org/10.1109/5254.708428

HINTON, Geoffrey E et ZEMEL, Richard S, 1994. Autoencoders, minimum description length and helmholtz free energy. Dans *Advances in neural information processing systems*, pages 3–10.

HOERL, Arthur E. et KENNARD, Robert W., 2000. Ridge regression : Biased estimation for nonorthogonal problems. *Technometrics*, 42 :80–86. doi :10.2307/1271436.
URL http://dx.doi.org/10.2307/1271436

HUANG, Furong, ASH, Jordan, LANGFORD, John et SCHAPIRE, Robert, 2017. Learning deep resnet blocks sequentially using boosting theory. *arXiv preprint arXiv :1706.04964*.

HUERTAS, Andres et NEVATIA, Ramakant, 1988. Detecting buildings in aerial images. *ICVGIP*, 41(2) :131–152.

HUNT, Bobby R, RYAN, Thomas W, SEMENTILLI, Philip J et DEKRUGER, D, 1991. Interactive tools for assisting the extraction of cartographic features. Dans *Image Understanding and the Man-Machine Interface III*, tome 1472, pages 208–219. International Society for Optics and Photonics.

Ienco, Dino, Pensa, Ruggero G et Meo, Rosa, 2012. From context to distance : Learning dissimilarity for categorical data clustering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(1) :1.

Igel, Christian, Hansen, Nikolaus et Roth, Stefan, 2007. Covariance matrix adaptation for multi-objective optimization. *Evolutionary computation*, 15(1) :1–28.

Ioffe, Sergey et Szegedy, Christian, 2015. Batch normalization : Accelerating deep network training by reducing internal covariate shift. Dans *International Conference on Machine Learning*, pages 448–456.

Isola, Phillip, Zhu, Jun-Yan, Zhou, Tinghui et Efros, Alexei A, 2016. Image-to-image translation with conditional adversarial networks. *arxiv*.

Jaccard, Paul, 1901. Distribution de la flore alpine dans le bassin des dranses et dans quelques régions voisines. *Bull. Soc. Vaud. Sci. Nat.*, 37 :241–272.

Jilani, Musfira, Corcoran, Padraig et Bertolotto, Michela, 2014. Automated highway tag assessment of openstreetmap road networks. Dans *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 449–452. ACM.

Jin, Yanghua, Zhang, Jiakai, Li, Minjun, Tian, Yingtao, Zhu, Huachun et Fang, Zhihao, 2017. Towards the automatic anime characters creation with generative adversarial networks. *arXiv preprint arXiv :1708.05509*.

Jindal, Ishan, Nokleby, Matthew et Chen, Xuewen, 2016. Learning deep networks from noisy labels with dropout regularization. Dans *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 967–972. IEEE.

Joachims, Thorsten, 1998. Making large-scale svm learning practical. Rapport technique, Technical Report, SFB 475 : Komplexitätsreduktion in Multivariaten Datenstrukturen, Universität Dortmund.

Jolliffe, Ian, 2011. Principal component analysis. Dans *International encyclopedia of statistical science*, pages 1094–1096. Springer.

Ju, Cheng, Bibaut, Aurélien et van der Laan, Mark J, 2017. The relative performance of ensemble methods with deep convolutional neural networks for image classification. *arXiv preprint arXiv :1704.01664*.

Kaiser, Pascal, Wegner, Jan Dirk, Lucchi, Aurélien, Jaggi, Martin, Hofmann, Thomas et Schindler, Konrad, 2017. Learning aerial image segmentation from online maps. *IEEE Transactions on Geoscience and Remote Sensing*, 55(11) :6054–6068.

Kamnitsas, Konstantinos, Bai, Wenjia, Ferrante, Enzo, McDonagh, Steven, Sinclair, Matthew, Pawlowski, Nick, Rajchl, Martin, Lee, Matthew, Kainz, Bernhard, Rueckert, Daniel *et al.*, 2017. Ensembles of multiple models and architectures for robust brain tumour segmentation. *arXiv preprint arXiv :1711.01468*.

KAMPFFMEYER, Michael, SALBERG, Arnt-Borre et JENSSEN, Robert, 2016. Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks. Dans *Proceedings CVPR Workshops*, pages 1–9.

KARRAS, Tero, AILA, Timo, LAINE, Samuli et LEHTINEN, Jaakko, 2017. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv :1710.10196*.

KASPI, Mor, RAVIV, Tal et TZUR, Michal, 2015a. Bike sharing systems : User dissatisfaction in the presence of unusable bicycles. Rapport technique, Working Paper, Tel-Aviv University.

KASPI, Mor, RAVIV, Tal et TZUR, Michal, 2015b. Detection of unusable bicycles in bike-sharing systems. *Omega*.

KHOSHELHAM, Kourosh, NARDINOCCHI, Carla, FRONTONI, Emanuele, MANCINI, Adriano et ZINGARETTI, Primo, 2010. Performance evaluation of automated approaches to building detection in multi-source aerial data. *ISPRS*, 65(1) :123–133.

KIEFER, Jack, WOLFOWITZ, Jacob *et al.*, 1952. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3) :462–466.

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

KIRKPATRICK, Scott, GELATT, C Daniel et VECCHI, Mario P, 1983. Optimization by simulated annealing. *science*, 220(4598) :671–680.

KITTLER, J., HATEF, M., DUIN, R. P. W. et MATAS, J., 1998. On combining classifiers. *IEEE PAMI*, 20(3) :226–239. doi :10.1109/34.667881.

KLEMA, Virginia et LAUB, Alan, 1980. The singular value decomposition : Its computation and some applications. *IEEE Transactions on automatic control*, 25(2) :164–176.

KOHONEN, Teuvo, 1997. Exploration of very large databases by self-organizing maps. Dans *Proceedings of International Conference on Neural Networks (ICNN'97)*, tome 1, pages PL1–PL6. IEEE.

KOU, Yufeng, LU, Chang-Tien et CHEN, Dechang, 2006. Spatial weighted outlier detection. Dans *Proceedings of the 2006 SIAM international conference on data mining*, pages 614–618. SIAM.

KRÄHENBÜHL, Philipp et KOLTUN, Vladlen, 2011. Efficient inference in fully connected crfs with gaussian edge potentials. Dans *Advances in neural information processing systems*, pages 109–117.

Krizhevsky, Alex et Hinton, Geoffrey, 2009. Learning multiple layers of features from tiny images.

Krizhevsky, Alex, Sutskever, Ilya et Hinton, Geoffrey E., 2012. Imagenet classification with deep convolutional neural networks. Dans *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105. Curran Associates Inc., USA.
URL http://dl.acm.org/citation.cfm?id=2999134.2999257

Lafferty, John D., McCallum, Andrew et Pereira, Fernando C. N., 2001. Conditional random fields : Probabilistic models for segmenting and labeling sequence data. pages 282–289.
URL http://dl.acm.org/citation.cfm?id=645530.655813

Lahiri, Avisek, Roy, Abhijit Guha, Sheet, Debdoot et Biswas, Prabir Kumar, 2016. Deep neural ensemble for retinal vessel segmentation in fundus images towards achieving label-free angiography. Dans *Engineering in Medicine and Biology Society (EMBC), 2016 IEEE 38th Annual International Conference of the*, pages 1340–1343. IEEE.

Larsen, Jan, Hansen, Lars Kai, Svarer, Claus et Ohlsson, M, 1996. Design and regularization of neural networks : the optimal use of a validation set. Dans *Neural Networks for Signal Processing [1996] VI. Proceedings of the 1996 IEEE Signal Processing Society Workshop*, pages 62–71. IEEE.

LeCun, Yann, Boser, Bernhard, Denker, John S, Henderson, Donnie, Howard, Richard E, Hubbard, Wayne et Jackel, Lawrence D, 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4) :541–551.

LeCun, Yann, Bottou, Léon, Bengio, Yoshua et Haffner, Patrick, 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324.

LeCun, Yann A, Bottou, Léon, Orr, Genevieve B et Müller, Klaus-Robert, 2012. Efficient backprop. Dans *Neural networks : Tricks of the trade*, pages 9–48. Springer.

Lemberger, Pirmin, Batty, Marc, Morel, MÃ©dÃ©ric et RaffaÃ«lli, Jean-Luc, 2015. *Big Data et Machine Learning – Manuel du data scientist*. Dunod.

Levitt, Stephen et Aghdasi, Farzin, 2000. Fuzzy representation and grouping in building detection. Dans *Image Processing, 2000.*, tome 3, pages 324–327. IEEE.

Li, Shuangjiang, Wang, Wei, Qi, Hairong, Ayhan, Bulent, Kwan, Chiman et Vance, Steven, 2015a. Low-rank tensor decomposition based anomaly detection for hyperspectral imagery. Dans *Image Processing (ICIP), 2015 IEEE International Conference on*, pages 4525–4529. Citeseer.

Li, Yexin, Zheng, Yu, Zhang, Huichu et Chen, Lei, 2015b. Traffic prediction in a bike-sharing system. Dans *SIGSPATIAL'15*.

Li, Yijun, Liu, Sifei, Yang, Jimei et Yang, Ming-Hsuan, 2017a. Generative face completion. Dans *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, tome 1, page 3.

Li, Yuncheng, Yang, Jianchao, Song, Yale, Cao, Liangliang, Luo, Jiebo et Li, Jia, 2017b. Learning from noisy labels with distillation. *CoRR, abs/1703.02391*.

Lin, Guosheng, Milan, Anton, Shen, Chunhua et Reid, Ian, 2016. Refinenet : Multi-path refinement networks with identity mappings for high-resolution semantic segmentation. *arXiv preprint arXiv :1611.06612*.

Liu, Wei, Anguelov, Dragomir, Erhan, Dumitru, Szegedy, Christian, Reed, Scott, Fu, Cheng-Yang et Berg, Alexander C, 2016. Ssd : Single shot multibox detector. Dans *European conference on computer vision*, pages 21–37. Springer.

Lloyd, Stuart, 1982. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2) :129–137.

Long, Jonathan, Shelhamer, Evan et Darrell, Trevor, 2015a. Fully convolutional networks for semantic segmentation. Dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440.

Long, Jonathan, Shelhamer, Evan et Darrell, Trevor, 2015b. Fully convolutional networks for semantic segmentation. Dans *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Lorensen, William E et Cline, Harvey E, 1987. Marching cubes : A high resolution 3d surface construction algorithm. Dans *ACM siggraph computer graphics*, tome 21, pages 163–169. ACM.

Lowe, David G, 1999. Object recognition from local scale-invariant features. Dans *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, tome 2, pages 1150–1157. Ieee.

Luan, Fujun, Paris, Sylvain, Shechtman, Eli et Bala, Kavita, 2017. Deep photo style transfer. *CoRR, abs/1703.07511*, 2.

Luo, Wenjie, Li, Yujia, Urtasun, Raquel et Zemel, Richard, 2016. Understanding the effective receptive field in deep convolutional neural networks. Dans *Advances in neural information processing systems*, pages 4898–4906.

Ma, Zhen, Tavares, João Manuel RS et Jorge, RM Natal, 2009. A review on the current segmentation algorithms for medical images. Dans *Proceedings of the 1st International Conference on Imaging Theory and Applications (IMAGAPP)*.

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

MacQueen, J., 1967. Some methods for classification and analysis of multivariate observations. Dans *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1 : Statistics*, pages 281–297.

MacQueen, James *et al.*, 1967. Some methods for classification and analysis of multivariate observations. Dans *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, tome 1, pages 281–297. Oakland, CA, USA.

Maggiori, Emmanuel, Charpiat, Guillaume, Tarabalka, Yuliya et Alliez, Pierre, 2017. Recurrent neural networks to correct satellite image classification maps. *IEEE TGRS*.

Marmanis, Dimitrios, Schindler, Konrad, Wegner, Jan Dirk, Galliani, Silvano, Datcu, Mihai et Stilla, Uwe, 2018. Classification with an edge : Improving semantic image segmentation with boundary detection.

Marmanis, Dimitrios, Wegner, Jan D, Galliani, Silvano, Schindler, Konrad, Datcu, Mihai et Stilla, Uwe, 2016. Semantic segmentation of aerial images with an ensemble of cnss. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 2016*, 3 :473–480.

McCallum, Andrew, Nigam, Kamal et Ungar, Lyle H, 2000. Efficient clustering of high-dimensional data sets with application to reference matching. Dans *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 169–178. ACM.

Meyer, Fernand, 1992. Color image segmentation. Dans *Image Processing and its Applications, 1992., International Conference on*, pages 303–306. IET.

Mille, Julien, Boné, Romuald, Makris, Pascal et Cardot, Hubert, 2006. Greedy algorithm and physics-based method for active contours and surfaces : a comparative study. Dans *Image Processing, 2006 IEEE International Conference on*, pages 1645–1648. IEEE.

Minsky, Papert, 1969. an introduction to computational geometry.

Mnih, Volodymyr, 2013. *Machine learning for aerial image labeling.* Thèse de doctorat, University of Toronto.

Mnih, Volodymyr et Hinton, Geoffrey E, 2012. Learning to label aerial images from noisy data. Dans *Proceedings of the 29th International conference on machine learning (ICML-12)*, pages 567–574.

Močkus, J, 1975. On bayesian methods for seeking the extremum. Dans *Optimization Techniques IFIP Technical Conference*, pages 400–404. Springer.

Moon, Todd K, 1996. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6) :47–60.

MOTIVATE, 2016a. Nycbs monthly operating reports. https://www.citibikenyc.com/system-data/operating-reports. Online ; accessed 15/02/2016.

MOTIVATE, 2016b. System data. https://www.citibikenyc.com/system-data. Online ; accessed 15/02/2016.

MÜLLER, Sönke et ZAUM, Daniel Wilhelm, 2005. Robust building detection in aerial images. *IAPRS*, 36(B2/W24) :143–148.

MÜNZ, Gerhard, LI, Sa et CARLE, Georg, 2007. Traffic anomaly detection using k-means clustering. Dans *GI/ITG Workshop MMBnet*, pages 13–14.

MUSICANT, David R, CHRISTENSEN, Janara M et OLSON, Jamie F, 2007. Supervised learning by training on aggregate outputs. Dans *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 252–261. IEEE.

NAPOLI, Bikesharing, 2013. Bike share boom : 7 cities doing it right. http://www.bikesharingnapoli.it/it/best-practices/. Online ; accessed 18/02/2016.

NASH, John, 1951. Non-cooperative games. *Annals of mathematics*, pages 286–295.

NATIONAL LIBRARY OF SCOTLAND, 1999. British first world war trench maps, 1915-1918. http://maps.nls.uk/ww1/trenches/info1.html. Accessed : 2018-09-21.

NECH, Aaron et KEMELMACHER-SHLIZERMAN, Ira, 2017. Level playing field for million scale face recognition. Dans *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3406–3415. IEEE.

NESTEROV, Yurii, 1983. A method for unconstrained convex minimization problem with the rate of convergence o (1/kˆ 2). Dans *Doklady AN USSR*, tome 269, pages 543–547.

NESTEROV, Yurii *et al.*, 2007. Gradient methods for minimizing composite objective function.

OYELADE, OJ, OLADIPUPO, OO et OBAGBUWA, IC, 2010. Application of k means clustering algorithm for prediction of students academic performance. *arXiv preprint arXiv :1002.2425*.

PARK, Mee Young et HASTIE, Trevor, 2007. L1-regularization path algorithm for generalized linear models. *Journal of the Royal Statistical Society : Series B (Statistical Methodology)*, 69(4) :659–677.

PASZKE, Adam, GROSS, Sam, CHINTALA, Soumith, CHANAN, Gregory, YANG, Edward, DEVITO, Zachary, LIN, Zeming, DESMAISON, Alban, ANTIGA, Luca et LERER, Adam, 2017. Automatic differentiation in pytorch. Dans *NIPS-W*.

PEARSON, Karl, 1901. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11) :559–572.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. et Duchesnay, E., 2011. Scikit-learn : Machine learning in Python. *Journal of Machine Learning Research*, 12 :2825–2830.

Peng, Chao, Zhang, Xiangyu, Yu, Gang, Luo, Guiming et Sun, Jian, 2017. Large kernel matters–improve semantic segmentation by global convolutional network. *arXiv preprint arXiv :1703.02719*.

Perrot, Alexandre, Bourqui, Romain, Hanusse, Nicolas, Lalanne, Frédéric et Auber, David, 2015. Large interactive visualization of density functions on big data infrastructure. Dans *Large Data Analysis and Visualization (LDAV), 2015 IEEE 5th Symposium on*, pages 99–106. IEEE.

Portnoy, Leonid, 2000. *Intrusion detection with unlabeled data using clustering*. Thèse de doctorat, Columbia University.

Pratt, Lorien Y, 1993. Discriminability-based transfer between neural networks. Dans *Advances in neural information processing systems*, pages 204–211.

Qucit, 2015. Bikepredict. https://www.qucit.com/bikepredict/. Online ; accessed 18/02/2016.

Raviv, Tal, Tzur, Michal et Forma, Iris A, 2013. Static repositioning in a bike-sharing system : models and solution approaches. *EURO Journal on Transportation and Logistics*, 2(3) :187–229.

Reed, Scott, Akata, Zeynep, Yan, Xinchen, Logeswaran, Lajanugen, Schiele, Bernt et Lee, Honglak, 2016. Generative adversarial text to image synthesis. *arXiv preprint arXiv :1605.05396*.

Reynolds, Douglas, 2015. Gaussian mixture models. *Encyclopedia of biometrics*, pages 827–832.

Riboni, Daniele, Bettini, Claudio, Civitarese, Gabriele, Janjua, Zaffar Haider et Helaoui, Rim, 2015. Fine-grained recognition of abnormal behaviors for early detection of mild cognitive impairment. Dans *Pervasive Computing and Communications (PerCom), 2015 IEEE International Conference on*, pages 149–154. IEEE.

Rieke, Roland, Zhdanova, Maria, Repp, Jürgen, Giot, Romain et Gaber, Chrystel, 2013. Fraud Detection in Mobile Payments Utilizing Process Behavior Analysis. Dans *The 2nd International Workshop on Recent Advances in Security Information and Event Management (RaSIEM 2013)*, page 8. France.
URL https://hal.archives-ouvertes.fr/hal-00841002

Robertson, David James et Burton, Anthony Michael, 2016. Unfamiliar face recognition : Security, surveillance and smartphones. *The Journal of the Homeland Defense and Security Information Analysis Center*, pages 14–21.

ROLNICK, David, VEIT, Andreas, BELONGIE, Serge et SHAVIT, Nir, 2017. Deep learning is robust to massive label noise. *arXiv preprint arXiv :1705.10694*.

RONNEBERGER, Olaf, FISCHER, Philipp et BROX, Thomas, 2015. U-net : Convolutional networks for biomedical image segmentation. Dans *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer.

ROSASCO, Lorenzo, VITO, Ernesto De, CAPONNETTO, Andrea, PIANA, Michele et VERRI, Alessandro, 2004. Are loss functions all the same ? *Neural Computation*, 16(5) :1063–1076.

ROSE, Gene, 2009. Pan sharpening. *Photogra. Engin. Remote Sens., September*, 17(2) :18–27.

ROSENBLATT, Frank, 1957. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory.

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

RUSSAKOVSKY, Olga, DENG, Jia, SU, Hao, KRAUSE, Jonathan, SATHEESH, Sanjeev, MA, Sean, HUANG, Zhiheng, KARPATHY, Andrej, KHOSLA, Aditya, BERNSTEIN, Michael, BERG, Alexander C. et FEI-FEI, Li, 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3) :211–252. doi :10.1007/s11263-015-0816-y.

SAMUEL, Arthur L, 1959. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3) :210–229.

SARWAR, Badrul M, KARYPIS, George, KONSTAN, Joseph et RIEDL, John, 2002. Recommender systems for large-scale e-commerce : Scalable neighborhood formation using clustering. Dans *Proceedings of the fifth international conference on computer and information technology*, tome 1, pages 291–324.

SASAKI, Yutaka *et al.*, 2007. The truth of the f-measure. *Teach Tutor mater*, 1(5) :1–5.

SCHAPIRE, Robert E, 1990. The strength of weak learnability. *Machine learning*, 5(2) :197–227.

SCHUBERT, Erich, ZIMEK, Arthur et KRIEGEL, Hans-Peter, 2014. Local outlier detection reconsidered : a generalized view on locality with applications to spatial, video, and network outlier detection. *Data Mining and Knowledge Discovery*, 28(1) :190–237. doi :10.1007/s10618-012-0300-z.
URL https://doi.org/10.1007/s10618-012-0300-z

SHAFER, Glenn, 1992. Dempster-shafer theory. *Encyclopedia of artificial intelligence*, pages 330–331.

Sharma, Kanishka, Rupprecht, Christian, Caroli, Anna, Aparicio, Maria Carolina, Remuzzi, Andrea, Baust, Maximilian et Navab, Nassir, 2017. Automatic segmentation of kidneys using deep learning for total kidney volume quantification in autosomal dominant polycystic kidney disease. *Scientific Reports*, 7.

Shin, Dear Sungbok, Choi, Minsuk, Choi, Jinho, Langevin, Scott, Bethune, Christopher, Horne, Philippe, Kronenfeld, Nathan, Kannan, Ramakrishnan, Drake, Barry, Park, Haesun *et al.*, 2017. Stexnmf : Spatio-temporally exclusive topic discovery for anomalous event detection. Dans *Data Mining (ICDM), 2017 IEEE International Conference on*, pages 435–444. IEEE.

Shotton, Jamie, Johnson, Matthew et Cipolla, Roberto, 2008. Semantic texton forests for image categorization and segmentation. Dans *Computer vision and pattern recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.

Shotton, Jamie, Sharp, Toby, Kipman, Alex, Fitzgibbon, Andrew, Finocchio, Mark, Blake, Andrew, Cook, Mat et Moore, Richard, 2013. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1) :116–124.

Simonyan, Karen et Zisserman, Andrew, 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv :1409.1556*.

Sirmacek, Beril et Unsalan, Cem, 2008. Building detection from aerial images using invariant color features and shadow information. Dans *ISCIS 2008.*, pages 1–5. IEEE.

Sirmacek, Beril et Unsalan, Cem, 2009. Urban-area and building detection using sift keypoints and graph theory. *IEEE Transactions on Geoscience and Remote Sensing*, 47(4) :1156–1167.

Sironi, Amos, Lepetit, Vincent et Fua, Pascal, 2014. Multiscale centerline detection by learning a scale-space distance transform. Dans *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2697–2704. IEEE.

Smith, Leslie N, 2017. Cyclical learning rates for training neural networks. Dans *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*, pages 464–472. IEEE.

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

Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya et Salakhutdinov, Ruslan, 2014. Dropout : A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15 :1929–1958.
URL http://jmlr.org/papers/v15/srivastava14a.html

Srivastava, Rupesh Kumar, Greff, Klaus et Schmidhuber, Jürgen, 2015. Highway networks. *arXiv preprint arXiv :1505.00387*.

Steinhaus, Hugo, 1956. Sur la division des corp materiels en parties. *Bull. Acad. Polon. Sci*, 1(804) :801.

Sukhbaatar, Sainbayar et Fergus, Rob, 2014. Learning from noisy labels with deep neural networks. *arXiv preprint arXiv :1406.2080*, 2(3) :4.

Susi, 2013a. How to be a good citi bike citizen. http://velojoy.com/2013/07/02/turned-bike-share-saddle/. Online ; accessed 18/02/2016.

Susi, 2013b. How to be a good citi bike citizen. https://velojoy.com/2013/07/02/turned-bike-share-saddle/. Online ; accessed 21/09/2018.

Sussillo, David, 2014. Random walks : Training very deep nonlin-ear feed-forward networks with smart ini. *arXiv preprint arXiv*, 1412.

Suykens, Johan AK et Vandewalle, Joos, 1999. Least squares support vector machine classifiers. *Neural processing letters*, 9(3) :293–300.

Svetnik, Vladimir, Liaw, Andy, Tong, Christopher, Culberson, J Christopher, Sheridan, Robert P et Feuston, Bradley P, 2003. Random forest : a classification and regression tool for compound classification and qsar modeling. *Journal of chemical information and computer sciences*, 43(6) :1947–1958.

Szegedy, Christian, Liu, Wei, Jia, Yangqing, Sermanet, Pierre, Reed, Scott, Anguelov, Dragomir, Erhan, Dumitru, Vanhoucke, Vincent et Rabinovich, Andrew, 2015. Going deeper with convolutions. Dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

Szegedy, Christian, Zaremba, Wojciech, Sutskever, Ilya, Bruna, Joan, Erhan, Dumitru, Goodfellow, Ian et Fergus, Rob, 2013. Intriguing properties of neural networks. *arXiv preprint arXiv :1312.6199*.

Tibshirani, Robert, 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288.

Törnros, T, Dorn, H, Hahmann, S et Zipf, A, 2015. Uncertainties of completeness measures in openstreetmap–a case study for buildings in a medium-sized german city. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 353–357.

Vakalopoulou, Maria, Karantzalos, Konstantinos, Komodakis, Nikos et Paragios, Nikos, 2015. Building detection in very high resolution multispectral data with deep learning features. Dans *IGARSS, 2015 IEEE International*, pages 1873–1876. IEEE.

van der Walt, Stéfan, Schönberger, Johannes L., Nunez-Iglesias, Juan, Boulogne, François, Warner, Joshua D., Yager, Neil, Gouillart, Emmanuelle, Yu, Tony et the scikit-image contributors, 2014. scikit-image : image processing in Python. *PeerJ*, 2 :e453. doi :10.7717/peerj.453.
URL http://dx.doi.org/10.7717/peerj.453

Veit, Andreas, Alldrin, Neil, Chechik, Gal, Krasin, Ivan, Gupta, Abhinav et Belongie, Serge, 2017. Learning from noisy large-scale datasets with minimal supervision. Dans *The Conference on Computer Vision and Pattern Recognition*.

Veit, Andreas, Wilber, Michael J et Belongie, Serge, 2016. Residual networks behave like ensembles of relatively shallow networks. Dans *Advances in Neural Information Processing Systems*, pages 550–558.

Vicente, Yago, Tomas, F, Hoai, Minh et Samaras, Dimitris, 2016. Noisy label recovery for shadow detection in unfamiliar domains. Dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3783–3792.

Viola, Paul et Jones, Michael, 2001. Rapid object detection using a boosted cascade of simple features. Dans *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, tome 1, pages I–I. IEEE.

Wang, Fei, Chen, Liren, Li, Cheng, Huang, Shiyao, Chen, Yanjie, Qian, Chen et Loy, Chen Change, 2018. The Devil of Face Recognition is in the Noise. *arXiv :1807.11649 [cs]*. ArXiv : 1807.11649.
URL http://arxiv.org/abs/1807.11649

Wang, Hongzhen, Wang, Ying, Zhang, Qian, Xiang, Shiming et Pan, Chunhong, 2017. Gated convolutional neural network for semantic segmentation in high-resolution images. *Remote Sensing*, 9(5) :446.

Wang, XY et Garibaldi, Jon M, 2005. A comparison of fuzzy and non-fuzzy clustering techniques in cancer diagnosis. Dans *Proceedings of the 2nd International Conference in Computational Intelligence in Medicine and Healthcare, BIOPATTERN Conference, Costa da Caparica, Lisbon, Portugal*, tome 28.

Wegner, Jan D, Branson, Steven, Hall, David, Schindler, Konrad et Perona, Pietro, 2016. Cataloging public objects using aerial and street-level images-urban trees. Dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6014–6023.

White, Tom, 2012. *Hadoop : The definitive guide*. " O'Reilly Media, Inc.".

Wolpert, David H, 1992. Stacked generalization. *Neural networks*, 5(2) :241–259.

Xie, Michael, Jean, Neal, Burke, Marshall, Lobell, David et Ermon, Stefano, 2015. Transfer learning from deep features for remote sensing and poverty mapping. *arXiv preprint arXiv :1510.00098*.

YAKIMOVSKY et FELDMAN, 2009. A semantics-based decision theory region analyser. *IJCAI*, 73 :580–588.

YANG, Lin, WU, Xiaqing, PRAUN, Emil et MA, Xiaoxu, 2009. Tree detection from aerial imagery. Dans *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 131–137. ACM.

YU, Fisher et KOLTUN, Vladlen, 2015. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv :1511.07122*.

YU, Zhiding et ZHANG, Cha, 2015. Image based static facial expression recognition with multiple deep network learning. Dans *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*, pages 435–442. ACM.

YUAN, Jiangye, 2016. Automatic building extraction in aerial scenes using convolutional networks. *arXiv preprint arXiv :1602.06564*.

YUAN, Jiangye, 2017. Learning building extraction in aerial scenes with convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*.

YUILLE, Alan L, HALLINAN, Peter W et COHEN, David S, 1992. Feature extraction from faces using deformable templates. *International journal of computer vision*, 8(2) :99–111.

ZAHARIA, Matei, CHOWDHURY, Mosharaf, FRANKLIN, Michael J, SHENKER, Scott et STOICA, Ion, 2010. Spark : Cluster computing with working sets. Dans *HotCLoud*, tome 10, pages 10–10.

ZENG, Nianyin, ZHANG, Hong, SONG, Baoye, LIU, Weibo, LI, Yurong et DOBAIE, Abdullah M, 2018. Facial expression recognition via learning deep sparse autoencoders. *Neurocomputing*, 273 :643–649.

ZHAO, Hengshuang, SHI, Jianping, QI, Xiaojuan, WANG, Xiaogang et JIA, Jiaya, 2016. Pyramid scene parsing network. *arXiv preprint arXiv :1612.01105*.

ZHAO, Kang, KANG, Jungwon, JUNG, Jaewook et SOHN, Gunho, 2018. Building extraction from satellite images using mask r-cnn with building boundary regularization. Dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 247–251.

ZIEGLER, Julius, BENDER, Philipp, SCHREIBER, Markus, LATEGAHN, Henning, STRAUSS, Tobias, STILLER, Christoph, DANG, Thao, FRANKE, Uwe, APPENRODT, Nils, KELLER, Christoph G *et al.*, 2014. Making bertha drive—an autonomous journey on a historic route. *IEEE Intelligent Transportation Systems Magazine*, 6(2) :8–20.

ZOJAJI, Zahra, ATANI, Reza Ebrahimi, MONADJEMI, Amir Hassan *et al.*, 2016. A survey of credit card fraud detection techniques : data and technique oriented perspective. *arXiv preprint arXiv :1611.06439*.