Techniques d'analyse de contenu appliquées à l'imagerie spatiale

Matthieu Le Goff

Université de Toulouse

THÈSE

En vue de l'obtention du

Doctorat de l'université de Toulouse

Délivré par :

Institut National Polytechnique de Toulouse (INP Toulouse)

Présentée et soutenue par :
Matthieu Le Goff
Le :
20/10/2017
Titre :

Techniques d'analyse de contenu appliquées aux images satellite

---

ED MITT : Domaine STIC Intelligence Artificielle

Unité de recherche :

Traitement et Compréhension des Images

Directeur(s) de Thèse :

Jean-Yves Tourneret

CoDT : Herwig Wendt

Rapporteurs :

Florence Tupin

Grégoire Mercier

Autre(s) membre(s) du jury :

Philippe Bolon
Jordi Inglada
Mathias Ortner
Marc Spigai

Chapitre 1 CONTENTS

LISTE DES FIGURES

LISTE DES TABLEAUX

REMERCIEMENTS

Pour la fin de cette enrichissante aventure, je tiens à remercier les personnes qui m'ont soutenu durant ces trois années de thèse et qui m'ont permis de mener à terme mon travail de recherche.

Dans un premier temps, je remercie mon directeur de thèse, Jean-Yves Tourneret, ainsi que mon codirecteur de thèse, Herwig Wendt, qui m'ont accompagné durant toute cette formation et ont su m'orienter dans mes recherches. Nos nombreux échanges constructifs m'ont toujours été profitables et m'ont permis de garder le cap.

En second lieu, je tiens à remercier Mathias Ortner et Marc Spigai pour avoir suivi avec intérêt mes travaux de recherche lors de nos nombreuses discussions et pour la bienveillance dont ils ont toujours fait preuve à mon égard. Les discussions techniques et les cafés du matin ont fortement contribué à la qualité de mes recherches et j'en ai tiré un immense bénéfice.

Je remercie également l'IRT Saint Exupéry pour le financement de cette thèse et l'opportunité de participer à une si belle aventure. Témoin des prémices de l'entreprise, je lui souhaite plein de réussite pour les prochaines années.

Je tiens aussi à remercier les deux rapporteurs : Mme. Florence Tupin et M. Grégoire Mercier pour avoir étudié et commenté avec pertinence mes travaux de recherche. Je remercie également M. Philippe Bolon et M. Jordi Inglada d'avoir accepté de prendre part au jury de ma thèse.

J'ai eu la chance d'effectuer dans le cadre de ce travail de recherche un séjour dans l'équipe de Data Science de Monash University à Melbourne. J'ai connu une expérience multiculturelle très enrichissante et je tiens à remercier tous les membres de l'équipe pour leur accueil et leur bienveillance à mon égard. En particulier, Geoff et Nayyar de m'avoir accordé du temps pour des discussions techniques passionnantes. Sans oublier François Petitjean qui en plus de m'avoir aidé dans mes travaux a rendu possible cette formidable expérience.

Je tiens à remercier l'ensemble de l'équipe OCE, en particulier Alicia, Caroline, Guillaume, Jonathan, Maxime et Maxime pour leur amitié et nos échanges divers. Petit clin d'œil à Mathieu Rouget qui m'a apporté beaucoup lors de mes premières expériences avec Spark.

Enfin, je tiens à remercier infiniment ma famille et ma belle-famille pour leur soutien et leur confiance. En particulier, ma femme Camille pour sa patience lors de ces trois dernières années.

INTRODUCTION GÉNÉRALE

Chapitre 2 INTRODUCTION GÉNÉRALE

2.1 CONTEXTE

Depuis les années 1970, la télédétection a permis d'améliorer l'analyse de la surface de la Terre grâce aux images satellites produites sous format numérique. Les images satellites apportent beaucoup d'informations complémentaires à des relevés physiques car elles ont une couverture spatiale plus importante et une période de revisite courte. L'essor de la télédétection a été accompagné de l'émergence des technologies de traitement qui ont permis aux utilisateurs de la communauté d'analyser les images satellites avec l'aide de chaînes de traitement de plus en plus automatiques.

La télédétection permet de mesurer l'impact de l'activité humaine sur l'environnement. Grâce à la télédétection, il est possible de déterminer la composition des sols, catégoriser les types de végétation sur un territoire, et cartographier les conséquences des activités humaines comme l'urbanisation. Pour effectuer cela, il est nécessaire de pouvoir analyser efficacement les images pour par exemple produire des cartes d'occupation des sols qui représentent une cartographie de types homogènes de milieux (zones urbaines, zones agricoles, forêts, ...) de la surface des terres émergées.

Les différentes missions d'observation de la Terre ont permis d'accumuler une quantité d'information importante dans le temps. Ceci est dû notamment à l'amélioration du temps de revisite des satellites pour une même région, au raffinement de la résolution spatiale et à l'augmentation de la fauchée (couverture spatiale d'une acquisition). La télédétection, autrefois appliquée à l'étude d'une seule image, s'est progressivement tournée et se tourne de plus en plus vers l'analyse de longues séries d'images multi-spectrales acquises à différentes dates. Le flux annuel d'images satellitaires est supposé atteindre plusieurs Pétaoctets dans les prochaines années.

La disponibilité d'une si grande quantité de données représente avant tout un atout pour développer des chaines de traitement avancées. Par ailleurs, les techniques d'apprentissage automatique beaucoup utilisées en télédétection se sont améliorées de manière significative. La robustesse des approches classiques d'apprentissage automatique étaient souvent limitées par la quantité de données disponibles pour la phase d'apprentissage. De nouvelles techniques ont été développées pour pouvoir utiliser efficacement ce nouveau flux important de données. Cependant, la quantité de données et la complexité des algorithmes mis en place nécessitent une grande puissance de calcul pour ces nouvelles chaînes de traitement.

En parallèle, la puissance de calcul accessible pour le traitement d'images s'est également accrue. Les GPUs (« Graphic Processing Unit ») sont de plus en plus utilisées et l'utilisation de cloud public ou privé est de plus en plus répandue. Désormais, pour le traitement d'images, toute la puissance nécessaire pour les chaînes de traitements automatiques est disponible à coût raisonnable. La conception des nouvelles chaînes de traitement doit prendre en compte ce nouveau facteur.

En télédétection, l'augmentation du volume de données à exploiter est devenue problématique due à la contrainte de la puissance de calcul nécessaire pour son analyse. Les algorithmes de télédétection traditionnels ont été conçus pour des données pouvant être stockées en mémoire interne tout au long des traitements. Cette condition est de moins en moins respectée avec la quantité croissante d'images à traiter et leur résolution. Les algorithmes de télédétection traditionnels nécessitent d'être revus et adaptés pour

le traitement de données à grande échelle. Ce besoin n'est pas propre à la télédétection et se retrouve dans d'autres secteurs comme le web, la médecine, la reconnaissance vocale… qui ont déjà résolu une partie de ces problèmes. Une partie des techniques et technologies développées par les autres domaines doivent encore être adaptées pour être appliquées au traitement d'images satellitaires.

Cette thèse se focalise sur l'étude d'algorithmes d'apprentissage appliqués à un volume massif d'images de télédétection. En particulier, un premier algorithme existant d'apprentissage automatique est étudié et adapté pour une implantation distribuée. L'objectif de l'implantation est le passage à l'échelle c'est-à-dire que l'algorithme puisse traiter une grande quantité de données moyennant une puissance de calcul adaptée. Enfin, la deuxième méthodologie proposée est basée sur des algorithmes récents d'apprentissage automatique, à savoir les réseaux de neurones convolutionnels qui sont adaptés à nos cas d'utilisation sur des images satellites.

2.2 CADRE DE LA THESE

Cette thèse s'est déroulée dans le cadre du projet OCE (Observation et Compréhension de l'Environnement) de l'IRT (Institute of Research and Technology) Saint Exupéry. Le but de ce projet au travers des différents axes de travail est d'améliorer l'accès à la donnée utile pour l'utilisateur. L'élaboration de chaînes de traitement automatiques de reconnaissance d'images satellite est alors un des axes principaux de travail de la partie dédiée au traitement d'images.

Les méthodes développées dans cette thèse ont été validées à partir d'images album basse résolution fournies par le satellite SPOT 6. La base de données mise à disposition par Airbus Defence and Space contenant plus de 200000 images offre une grande diversité de conditions d'acquisition et de paysages essentiels pour une évaluation correcte des algorithmes.

2.3 PLAN

Le manuscrit est structuré de la manière suivante :

- Le Chapitre 3 introduit les principaux éléments de contexte. Un constat sur les changements actuels du monde de l'imagerie spatiale est réalisé. Les bouleversements technologiques responsables de ces changements sont présentés et en particulier, un état de l'art des techniques d'apprentissage est détaillé.
- Dans le Chapitre 4, une revue des méthodes de passage à l'échelle d'une méthode d'apprentissage particulière est réalisée. Une nouvelle méthodologie est alors proposée après l'analyse des avantages et inconvénients des méthodes existantes.
- Le Chapitre 5 propose une application des technologies récentes d'apprentissage (les réseaux convolutionnels) aux images spatiales. Une nouvelle formulation est tout d'abord proposée pour leur application. Une analyse de l'application de ces techniques dans d'autres domaines a conduit à la conception de la chaîne de traitement proposée dans ce chapitre.

2.4 CONTRIBUTIONS

Les principales contributions de cette thèse sont listées ci-dessous :

Chapitre 3 : Une nouvelle formulation d'un algorithme distribué de Boosting est proposée. Sa conception met l'accent sur ses propriétés de passage à l'échelle tout en conservant les propriétés essentielles et intéressantes du Boosting qui manquaient aux algorithmes existants.

Chapitre 4 : Une méthodologie d'application de réseaux convolutifs pour la détection de nuages et la classification de terrain dans des images satellitaires est proposée. Aucune méthodologie clairement établie n'existait alors pour l'application de réseaux de neurones profonds aux images satellites et notre méthodologie formalise les opérations nécessaires pour leur apprentissage incluant la formulation du problème de classification et les prétraitements de données. En raison de la physionomie du problème, un patron de structure du réseau de neurones est proposé et ses paramètres seront optimisés lors des expériences.

LISTE DES PUBLICATIONS

CONFÉRENCES INTERNATIONALES

Le Goff M., Tourneret J.-Y., Wendt H., Ortner M., and Spigai M., “Distributed boosting for cloud detection” in Proc. of IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Beijing, China, 2016, pp. 2626–2629.

Le Goff M., Tourneret J.-Y., Wendt H., Ortner M., and Spigai M., “Deep Learning for Cloud Detection” in Proc. of International Conference on Pattern Recognition Systems (ICPRS), Madrid, Spain, 2017.

ETAT DE L'ART

Chapitre 3 ETAT DE L'ART

3.1 CONTEXTE

3.1.1 Imagerie satellite

Tous les jours, des satellites prennent des photos de la Terre depuis l'espace. Les images satellites ainsi produites frappent d'abord par leur beauté : elles offrent des paysages et des perspectives jusque-là inconnus à l'homme. Cependant, toutes ces images sont également utilisées pour un vaste panel applications. En réalité, l'analyse et la vente d'images spatiales sont devenues le cœur de métier de certaines entreprises telles que Spaceknow, Terra Bella, Airbus GeoInformation et Telespazio. Les images satellites sont de plus en plus utilisées et diffusées par les acteurs du spatial : par exemple, Airbus Defence and Space et Thales Alenia Space pour les acteurs européens mais également des acteurs américains avec Digital Globe ou Planet Labs.

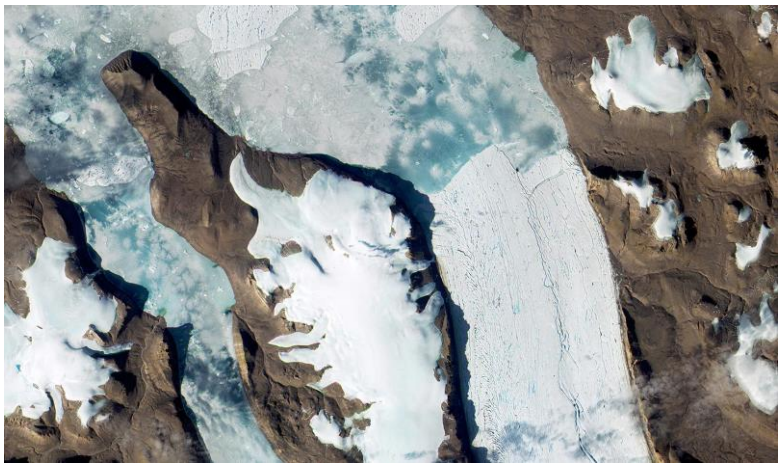

Figure 1: Image du Groenland par Sentinel 2. © ESA

Historiquement, les images satellites ont été souvent utilisées pour des missions militaires et les applications grand public sont restées des applications de niche. Les dernières avancées technologiques permettent encore aujourd'hui de découvrir de nouvelles applications. Les images satellites sont surtout utilisées pour observer des phénomènes à très grande échelle : en effet, une prise de vue satellite permet d'observer une surface de l'ordre du plusieurs kilomètres voire une dizaine de kilomètres de large. Elles sont souvent plus économiques et plus efficaces qu'une étude sur le terrain par exemple. Leur utilisation permet principalement de faire de la surveillance ou du suivi. L'agriculture est par exemple un domaine qui profite des avancées dans le domaine de l'imagerie spatiale. Les images spatiales permettent aujourd'hui de surveiller les plantations à l'échelle d'un pays entier de manière efficace et peu couteuse comparativement à la surface à surveiller. Cette surveillance a plusieurs buts dont la prédiction des récoltes (Figure 2) pour les marchés de matière première, la surveillance et la prévision de pénurie. De manière beaucoup plus ponctuelle, les images permettent également d'estimer les états des cultures pour les agriculteurs. Les images satellites pourraient permettre de répondre à bien d'autres questions et besoins. Les images satellites sont utilisées pour déterminer le niveau d'activité d'une usine automobile en calculant le taux d'occupation de ses parkings. Les images satellites peuvent être également utilisées

pour mesure des flux. Par exemple, le niveau d'activité d'un aéroport pourrait être estimé en calculant le nombre d'avions sur les tarmacs quotidiennement ou encore le niveau d'activité d'un port en comptant les bateaux. Les images satellites sont utilisées pour mesurer et surveiller des phénomènes à grande échelle comme l'expansion des villes, la fonte des glaces, ou la déforestation. Les images satellites sont très utiles pour évaluer la nature et l'occupation des sols (Figure 3) et sont par exemple beaucoup utilisées en agriculture pour vérifier le respect de réglementation telle que la politique agricole commune européenne. Pour conclure, dans le domaine du renseignement, les images satellites sont principalement utilisées pour surveiller l'activité militaire d'une zone sensible.

La recherche dans ce domaine est en pleine croissance et une grande partie des réponses est déjà connue mais il reste du chemin avant de voir de tels produits en production. La principale raison reste que tous ces services sont aujourd'hui possibles grâce à des avancées récentes dans certains domaines technologiques.

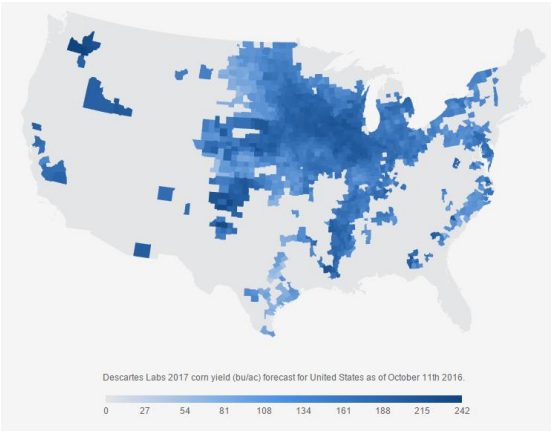

Figure 2: Estimation de la production de maïs aux Etats Unis à l'aide d'images satellite. © Descartes Labs

Le premier facteur qui a permis cette expansion est la disponibilité des images spatiales. En effet, des grandes bases de données d'images spatiales sont nécessaires pour développer des services qui sont, pour la plupart, basés sur l'analyse d'archives d'images. Les analyses peuvent nécessiter des images diverses du monde entier, de même que des séries temporelles (séries d'images du même endroit). Ce facteur fut longtemps un frein pour l'analyse d'images mais le nombre croissant de lancement de satellites d'observation et la simplification de l'accès à l'image ont permis au grand public et aux chercheurs d'accéder à plus d'images. De la même manière, la plupart des grands producteurs d'images ont orienté leur stratégie vers la production massive d'images. Dans ce cadre-là, le programme européen Copernicus a été lancé pour développer les services liés aux données spatiales. Copernicus est le nom d'un « programme européen de surveillance de la Terre ». Le programme Copernicus a pour but de rassembler l'ensemble des données obtenues à partir de satellites environnementaux et d'instruments de mesure sur site, afin de produire une vue globale et complète de l'état de notre planète. De cette manière, il sera possible de disposer d'informations et de services fiables chaque fois que cela est nécessaire pour la construction de futurs services. Il s'agit d'une initiative conjointe de l'Agence Spatiale Européenne (ESA) et de l'Union Européenne, qui vise à doter l'Europe d'une capacité opérationnelle et autonome d'observation de la Terre en « accès libre, plein et entier ». De manière pratique, le programme Copernicus s'appuie sur des satellites de la famille SENTINEL. Le satellite SENTINEL 1-A (satellite radar) fut lancé en 2014 et SENTINEL 1-B a été lancé en 2015. De la même manière, les satellites optiques SENTINEL 2-A et 2-B ont été lancés respectivement en 2015 et 2016. Ces 4 satellites sont aujourd'hui en vol et continuent

d'acquérir des images pour le catalogue de Copernicus. Les satellites S2 ont été calibrés afin de produire une image de la Terre entière tous les 5 jours. De la même manière, les producteurs privés d'images ont lancé leur propre constellation afin de pallier le besoin du marché en termes d'abondance d'images. Airbus Defence and Space et son programme AstroTerra (programme incluant les manœuvres des satellites SPOT 6-7 et Pléiades 1a-b) produisent l'équivalent en images de plusieurs fois la surface terrestre par an. De la même manière, l'acteur américain Planet possède également une flotte de satellites excédant les 60 satellites. Le nombre de satellites des différentes constellations permet de garantir un taux de couverture, de revisite et une diversité de conditions d'acquisitions. De manière moins impressionnante, Digital Globe possède également sa constellation des WorldViews composée de 5 satellites très haute résolution et une archive importante à disposition.

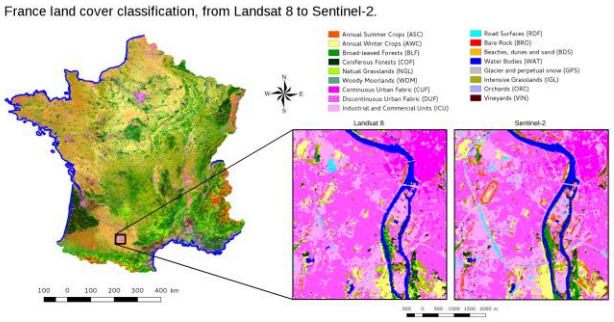

Figure 3: Occupation des sols à l'échelle de la France © CESBIO

Le monde de l'imagerie spatiale est en pleine révolution car des avancées technologies dans des domaines connexes bouleversent son fonctionnement historique. L'utilisation historique des images satellites semblent désormais dépassée : des services qui nécessitaient la commande d'une image particulière et l'analyse par un photo-interprète semblent être voués à être remplacés par l'utilisation des images de la zone en question dans les divers catalogues et l'analyse automatique à grande échelle. Par exemple, la carte d'occupation des sols (comme Figure 3) a été obtenue grâce à la disponibilité des images et de la puissance de calcul. Son calcul aurait été impossible avec l'approche historique. Les principaux facteurs d'accélération sont la disponibilité des images et le développement de l'informatique « en nuage » qui permet de mobiliser d'importantes capacités de calcul à moindre coût. Et enfin, il est désormais possible de faire reconnaître à un ordinateur ce qu'il voit grâce aux dernières avancées en reconnaissance d'images.

3.1.2 Une révolution

La révolution numérique a un impact sur le monde de la recherche incluant la recherche dans le domaine de l'imagerie spatiale. La révolution numérique est caractérisée par l'explosion des puissances de calcul à disposition du grand public, l'accès à la donnée à grande échelle et enfin des progrès importants en intelligence artificielle (en particulier en apprentissage automatique (Machine Learning) et apprentissage profond (Deep Learning)). La puissance de calcul a longtemps été un frein à la recherche mais l'apparition de nouveaux composants tels que les processeurs graphiques (GPUs[1]) ou encore l'accès au cloud public

[1] Graphic Processing Units

facilite l'accès à une puissance de calcul importante à coûts réduits. Les sources de données se sont multipliées : des initiatives d'ouverture de données et les compétitions d'analyse de données ont mis à disposition des analystes des données et des problèmes à résoudre. La dernière composante est l'apprentissage automatique qui consiste à générer des programmes informatiques reproduisant un comportement spécifié à l'avance : c'est effectivement dans ce cadre-là que l'on peut apprendre à l'ordinateur à reconnaitre certains objets. En effet, grâce à une série d'exemple d'images contenant l'objet en question, un programme peut « apprendre » à reconnaitre cet objet dans de nouvelles images. Ce domaine connaît aujourd'hui un développement sans égal grâce à des années de recherche en mathématiques et informatique mais également à l'accès à la puissance de calcul et l'accès aux données. Des programmes informatiques surpassent actuellement les humains sur des jeux tels que les échecs ou le jeu de go. Effectivement, les exemples de DeepBlue et Gasparov aux échecs ou encore AlphaGo et Lee Sedol plus récemment ont prouvé que les programmes d'intelligence artificiels actuels ont surpassé l'intelligence humaine sur certaines tâches très spécifiques.

La révolution numérique a également raccourci le cycle de recherche : les équipes de recherche sont de plus en plus liées avec les équipes de production et leur besoin. La mise en production des résultats de recherche est beaucoup plus rapide grâce à de nouveaux outils informatiques. Les problèmes et leurs approches pour les résoudre changent. En effet, l'aboutissement de l'intelligence artificielle consistant à générer un programme par mimétisme est l'automatisation de tâches. Une partie des problèmes historiques sont en train d'être résolus par l'intelligence artificielle et la puissance de calcul. Tout problème de Machine Learning en particulier peut se résumer à un problème d'optimisation : l'approche consistant à mettre plus de puissance de calcul pour la résolution du problème s'est peu à peu imposée sur l'approche subtile consistant à trouver une nouvelle formulation qui permettrait d'améliorer la convergence. Ainsi, une formulation du problème sous forme de problème d'apprentissage est souvent suffisante pour résoudre le problème sous réserve d'avoir la puissance de calcul adéquate. Les récentes avancées ont largement contribué à l'automatisation de résolution de ce type de problèmes.

L'apprentissage automatique est soutenu par un grand nombre d'acteurs privés qui le considère comme une ressource stratégique. Leur soutien se manifeste par le financement de recherches et le maintien et développement de librairies open source. Ces acteurs sont des acteurs qui ont accès simultanément à la quantité de données, aux problèmes à résoudre et à la puissance de calcul. Les grands acteurs du Web que sont Google, Amazon, Facebook, et Microsoft font partie de ces acteurs qui ont une place centrale et privilégiée dans cette nouvelle ère. En particulier, les fournisseurs de puissance de calcul (autrement appelés « cloud providers » car ils profitent de l'architecture en nuage pour le faire) que sont Google, Amazon et Microsoft ont déjà une place centrale pour l'accès à la puissance de calcul. Un fournisseur de puissance de calcul est une entité qui va permettre à un utilisateur de louer des machines autrement dit de la puissance de calcul. La facilité d'accès à la puissance de calcul a pour conséquence que la ressource fondamentale est désormais la donnée qui constitue alors une ressource stratégique.

Le projet OCE s'inscrit dans la volonté d'aider les partenaires industriels qui sont des acteurs du domaine du spatial et, en particulier, de l'imagerie spatiale, à s'adapter à cette révolution. En particulier, cette révolution s'accompagne d'une conséquence à double tranchant : la puissance de calcul associée à l'intelligence artificielle offre de toutes nouvelles perspectives mais permet également au grand public de développer des services basés de l'image satellite. La diminution du coût des images et la mise à disposition d'images satellite gratuitement fait que la barrière à l'entrée du marché de l'image satellite est beaucoup moins importante qu'auparavant. Historiquement, les producteurs d'images satellites ont

toujours profité d'un savoir-faire qu'ils étaient les seuls à détenir : cette révolution est donc accompagnée d'une nouvelle concurrence. Le but du projet OCE est de prendre en main les différentes technologies évoquées qui ne sont pas encore totalement diffusés dans ce domaine, et de les évaluer. En d'autres termes, le but est de faire passer les technologies du cloud et d'apprentissage d'un TRL 3 vers un TRL 6 (voir Figure 4).

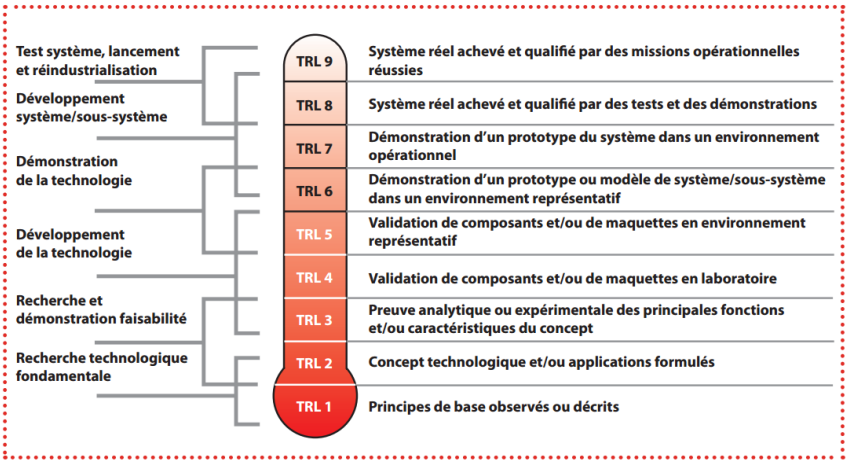

Figure 4: Echelle des TRLs utilisés pour qualifier le niveau de maturité d'une recherche. Source

3.1.3 Machine Learning

Ces dernières années ont été particulièrement fructueuses en termes d'avancées pour le monde du Machine Learning ou apprentissage automatique. Cependant, la recherche dans le domaine est encore très active : les techniques changent très vite et ne sont pas encore stabilisées. Cependant, les fondamentaux sur lesquels se basent ces techniques sont connus depuis longtemps avec par exemple le Neocognitron qui date de 1988 (Fukushima 1988). Les dernières avancées ont été permises par l'accessibilité à la puissance de calcul car les méthodes ont toujours été consommatrices de temps de calcul et par la disponibilité de la donnée qui a permis de construire des benchmarks entre les différentes méthodes. Il faut cependant quelques points de contexte pour mieux appréhender le changement qui s'opère.

Le domaine du Machine Learning ou apprentissage automatique repose sur des techniques qui ont besoin de données à analyser pour fonctionner. Pendant beaucoup d'années, le manque de données a été un frein à la recherche. La mise à disposition de jeux de données gratuites a largement contribué à la diffusion du Machine Learning. Des jeux de données gratuits pour des compétitions telles celles proposées par Kaggle ou pour comparer les différents algorithmes comme Imagenet (Deng J., et al. 2009), CIFAR ou MNIST ont stimulé la recherche et introduit de nouveaux utilisateurs dans la communauté (Figure 8). Le lien entre ses compétitions et l'avancée de la recherche est mis en évidence par le fait que jusqu'en 2012, les Random Forests (Breiman L., Random forests 2001) remportaient couramment les compétitions mais l'année 2012 a également été marqué par les premiers réseaux de neurones profonds vainqueurs de compétition (Large Scale Visual Recognition Challenge LSVRC (Krizhevsky A., Sutskever I. et Hinton G. 2012)). Aujourd'hui, plus de 50% des compétitions sont remportées par des réseaux de neurones. Les compétitions ont toujours permis de mettre en valeur l'état de l'art de la recherche dans leurs domaines respectifs. Chaque compétition a permis de faire évoluer les techniques et ces dernières années ont également vu apparaître et presque disparaître un grand nombre de techniques :

- Pretraining (plus utilisé)
- Autoencoder
- Dropout
- Optimisers : Momentums, RMSProp, Adam

Le monde de la recherche grâce à sa grande communauté dynamique et l'aspect ludique des compétitions est extrêmement actif : la taille de la communauté est en croissance exponentielle (Figure 5 et Figure 8). Une question se pose alors : quel sont les facteurs qui peuvent expliquer cet attrait soudain ? L'accès aux travaux de la communauté est essentiel pour le partage, la rationalisation et le développement de la recherche : la communauté est particulièrement tournée vers l'open source. L'open source explique la diffusion et l'adoption rapide des dernières recherches sur le sujet par la communauté en particulier pour les techniques d'apprentissage automatique et d'apprentissage profond. En effet, les découvertes sont souvent diffusées à la communauté sous forme de programmes de sorte qu'elles soient tout de suite appliquées à des cas réels. Le second facteur est l'accès à des outils performants en termes de programmation et d'exécution : l'open source a également contribué au développement rapide de libraires d'apprentissage (Figure 7). Ces librairies (Tensorflow, Theano, Mxnet, Scikit-learn…) contribuent en grande partie au développement de la recherche sur le sujet. Elles ont permis la diffusion d'outils complexes tels que la différenciation automatique, ou le calcul automatique du gradient. La qualité des outils à disposition est étroitement liée à la vitesse de la recherche : ils améliorent considérablement le temps de développement et de prototypage. De plus, les principaux acteurs privés évoqués précédemment investissent massivement dans ce domaine : en particulier, ils maintiennent leur librairie en open source dans l'objectif de profiter des retours utilisateurs pour améliorer leur produit. Les fournisseurs de cloud commencent à proposer des services dédiés à l'apprentissage. La plupart de ces techniques sont très consommatrices de calcul et l'accessibilité à de la puissance de calcul (Cloud, GPUs) est le dernier facteur qui permet ce développement exponentiel. En conclusion, les principaux facteurs de croissance sont d'avantage liés à l'ingénierie : d'où la conclusion générale sur cette croissance, son succès est d'avantage lié à une avancée en termes d'ingénierie.

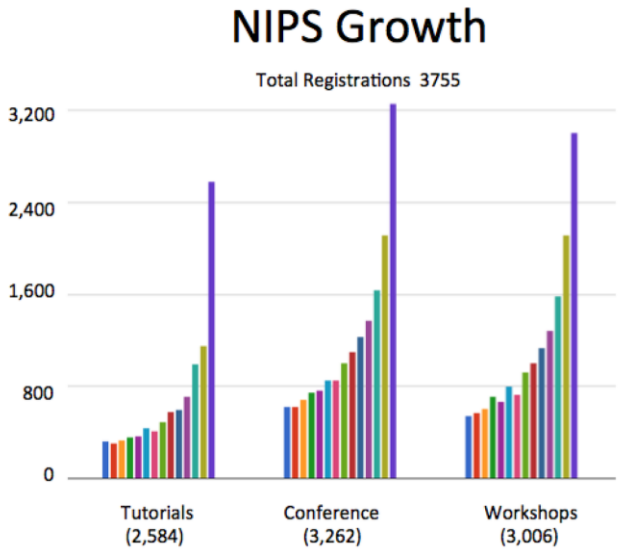

Figure 5: Nombre d'enregistrements pour la conférence NIPS en croissance exponentielle. Source

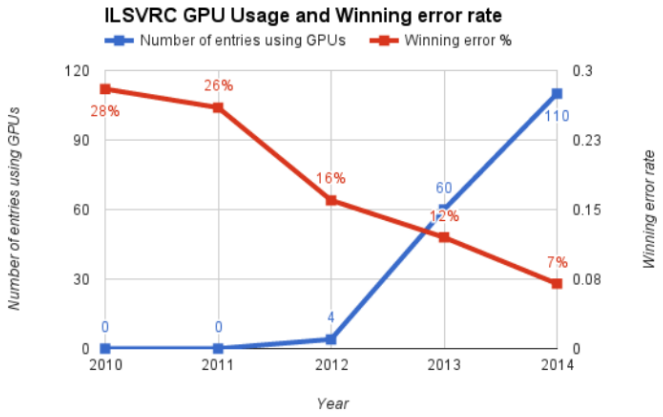

Figure 6: Nombrée d'entrées utilisant des GPUs au challenge ILSVRC. Source

Figure 7: Principales librairies open source de deep learning existantes en 2017. Source.

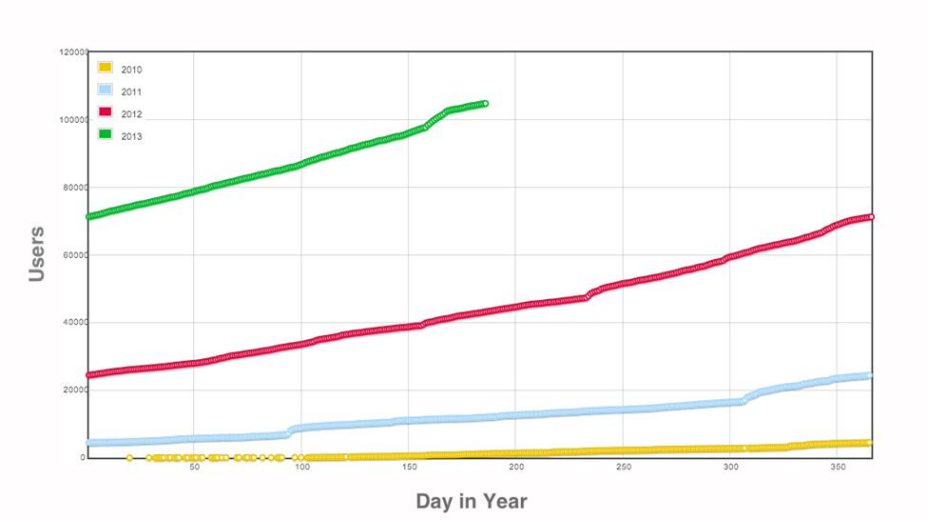

Figure 8: Evolution du nombre de comptes Kaggle en fonction du temps. © Kaggle

3.1.4 Comment utiliser ces nouveaux outils sur des taches relativement anciennes ?

Tous ces changements soulèvent la question suivante : quels sont les impacts de ces technologies sur des applications historiques et en particulier, quelle sera l'interaction entre les nouvelles approches et des

approches historiques. Les compétitions et les jeux de données à disposition ont orienté les applications et les développements vers le développement de modèles à la recherche des meilleures performances sur un problème fixé sans se soucier de leur application finale. En effet, les compétitions sont organisées de façon à classer les participants par rapport à une certaine métrique et il est donc naturel que les problèmes de classification et de régression aient été beaucoup traités. L'application des techniques d'apprentissage pour la résolution de problèmes historiques comme par exemple, celui de la classification de nuages, est un sujet de recherche actif. La plupart des domaines possèdent leurs contraintes métier et la flexibilité de ces nouveaux outils reste à démontrer même si leur réussite sur les cas d'utilisation de certaines entreprises comme Google pousse à croire qu'ils révolutionneront également un grand nombre de domaines d'application. L'utilité de ces nouveaux outils est l'automatisation de tâches comme la détection de nuages ou la classification des sols. Ces nouveaux outils permettent d'automatiser par la machine une tâche à partir de deux principales composantes : une base de données d'exemples et une métrique pour calculer la performance de la machine ou de l'utilisateur. Auparavant, l'automatisation d'une tâche était plus complexe, demandait à une équipe de recherche de définir et d'analyser un problème et demandait quelques années de recherche. Cette automatisation soulève beaucoup d'interrogations : quelle sera la performance de ces nouveaux outils comparativement aux outils développés avec l'approche historique qui sont actuellement utilisés et maintenus depuis plusieurs dizaines d'années ?

3.2 NOS DEUX EXEMPLES DE TRAVAIL

3.2.1 La détection de nuages

Comme présenté précédemment, les images satellites sont utilisées afin d'en extraire de l'information comme la nature des champs de culture, la nature du sol, ou la présence d'objets. Un des principaux problèmes des images optiques est la présence de nuages sur les images. La présence des nuages doit être indiquée dans les métas données d'une image au même titre que l'image est corrigée géométriquement ou radiométriquement. En effet, les pixels contenant des nuages doivent être évités lors des corrections atmosphériques, de la classification des sols, de la détection de changement ou de l'inversion de modèles biophysiques. En pratique, un masque de nuages est fourni avec l'image satellite : les pixels contenant des nuages sont indiqués en blanc par exemple sur le masque de nuages (Figure 9). D'un point de vue plus macroscopique, la couverture nuageuse d'une image a un grand impact sur la qualité et par conséquence sur le prix de l'image. De même, lors d'une commande spécifique d'un client, l'image peut être reprogrammée si l'image acquise contient trop de nuages.

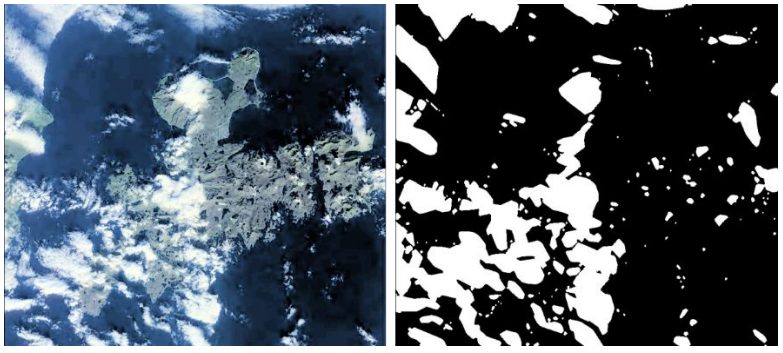

Figure 9: Exemple d'une image et de son masque de nuages associé.

Le problème et le sujet de recherche sont donc importants à en juger par les efforts du CNES et de l'ESA entre autres (Latry C., Panem C. et Dejean P. 2007) (Hagolle O., et al. 2010) (Hollstein A., et al. 2016). Le problème semble facile d'un premier abord : selon certaines conditions, les nuages sont détectables puisque les spectres du nuage et de l'eau par exemple ne se chevauchent pas. Cependant, il peut être difficile de détecter les nuages dans certains cas : des nuages fins vont être difficiles à différencier de l'arrière-plan (Figure 12). De même, il existe une grande variété de nuages en termes de formes, épaisseurs ou textures (Figure 10, Figure 11, Figure 12). Certains paysages ont des propriétés spectrales extrêmement proches du nuage : la neige est sans doute le cas connu le plus difficile mais les sables en bord de mer et les villes dans une moindre mesure peuvent être confondus avec un nuage (Figure 13).

Figure 10: Exemple d'une image avec beaucoup de nuages.

Figure 11: Exemple d'images contenant quelques nuages.

Figure 12: Exemple d'une image contenant de la brume.

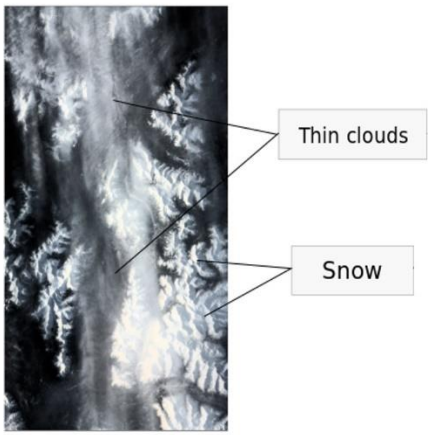

Figure 13: Exemple d'un nuage avec neige et nuages. La distinction est difficile.

Le problème a déjà été étudié et il existe une grande quantité de méthodes de détection de nuages. Cependant, les approches actuelles induisent une forte dépendance avec le satellite et bien souvent, un manque de robustesse est observé. L'idée principale est d'utiliser les méthodes automatiques pour établir une méthodologie robuste indépendante du satellite (du moins, qu'elle puisse se réappliquer à un autre satellite sans trop de difficultés).

3.2.2 La classification des sols.

Les images satellites sont utilisées afin d'en extraire des informations, par exemple, des informations sur les cultures pour l'agriculture. L'information précise de la nature des sols est donc une information essentielle pour ce type d'applications. Cette information permet de mesurer des indices environnementaux et de mesurer l'impact de la société sur son environnement. De manière générale, cette information est utilisée pour la surveillance de l'environnement, l'agriculture, la foresterie, le transport, la réponse aux catastrophes naturelles, … L'évolution et les changements de la nature des sols sont suivis avec attention. Il existe de nombreuses initiatives de génération de cartes d'occupation des sols (Figure 3 et (Broxton P. 2014)). Le besoin d'une meilleure estimation de couverts s'est déjà fait ressentir et des actions pour la construire ont été prise par l'union Européenne en autres (voir Group on Earth Observation (GEO) et Global Earth Observation System of Systems (GEOSS)). Les programmes de génération de cartographie (Figure 3, Figure 14, Figure 15) ont pour objectifs :

- Biodiversité : comprendre, surveiller et conserver la biodiversité
- Agriculture : soutenir l'agriculture durable et lutter contre la désertification
- Écosystèmes : amélioration de la gestion et de la protection des écosystèmes terrestres, côtiers et marins
- Météo : améliorer les informations météorologiques et les prévisions
- Eau : améliorer la gestion des ressources en eau grâce à une meilleure compréhension du cycle de l'eau
- Climat : comprendre, évaluer, prévoir, atténuer et s'adapter à la variabilité et au changement climatique
- Énergie : amélioration de la gestion des ressources énergétiques

- Santé : compréhension des facteurs environnementaux affectant la santé humaine et le bien-être
- Catastrophes : réduction de la perte de vie et des biens causés par des catastrophes naturelles et provoquées par l'homme

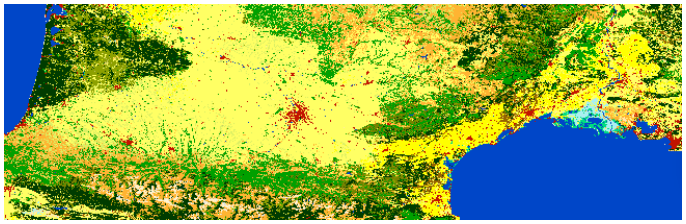

Figure 14: Extraction du sud de la France de carte de couvertures des sols maintenus par l'ESA. ©ESA

Quelques-unes de ces cartes sont déjà en accès libre comme celle représentée sur la Figure 15. L'ESA a conduit plusieurs missions de récolte de données pour la génération de telles cartes avec par exemple la Corinne Land Cover mise à jour avec les données du programme Copernicus. Cette carte permet d'avoir l'information de la répartition de plus de 40 types de terrain sur le territoire européen. La Land Cover utilisée ici (Figure 15, Figure 16) produite par USGS[2] est une carte des sols incluant une quinzaine de catégories dont les données ont été récoltées de 2001-2010 (Broxton P. 2014). La carte est d'une faible précision de l'ordre du kilomètre carré et d'un degré de confiance assez faible puisque les données ont été récoltées sur 10 ans et qu'il y a maintenant un peu moins d'une dizaine d'années qu'elle n'a pas été mise à jour. Cependant, une grande partie de la carte reste encore juste aujourd'hui et peut servir à généraliser l'information sur les parties manquantes.

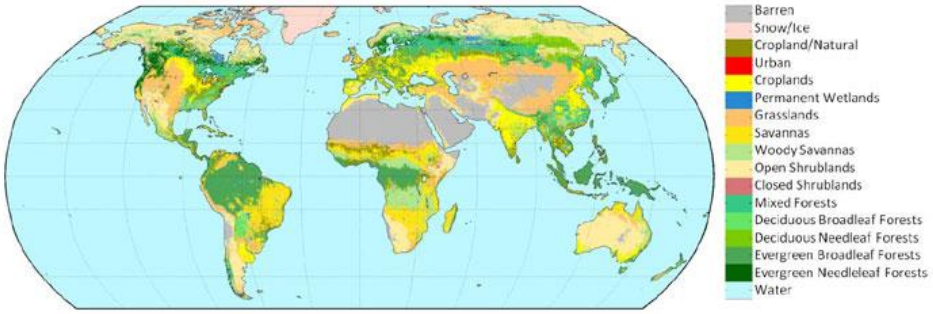

Figure 15: Carte de couverture des sols utilisée dans cette étude. ©USGS

[2] https://landcover.usgs.gov/global_climatology.php

Couleur	Légende
	Nuages
	Terrain aride
	Neige/Glace
	Champs en friche/Naturel
	Terrain urbain
	Champs (agriculture)
	Terrain marécageux
	Prairies
	Savanes
	Savanes boisées
	Fruticée dense
	Fruticée aérée
	Mélange de forêts
	Forêt tempérée décidues (formée d'arbres à feuilles caduques))
	Forêt de conifères décidus (forêt de résineux)
	Forêt tempérée sempervirente à feuilles caduque
	Forêt tempérée sempervirente de résineux
	Eau

Figure 16: Légende de la carte de couverture des sols

3.2.3 Panorama des acteurs

La détection de nuages est un domaine actif de recherche pour la plupart des grands acteurs. Une littérature extensive existe sur le sujet : la plupart de ces recherches sont financées par des acteurs privés. Le CNES et l'ESA ont beaucoup travaillé sur le sujet en prévision du segment sol de SENTINEL (Hagolle O., et al. 2010) (Latry C., Panem C. et Dejean P. 2007). Les acteurs américains tels que Planets ont également publié des travaux sur le sujet avec en particulier des approches avancées en termes d'apprentissage automatique[3]. Les recherches ont déjà commencé depuis quelques années à se concentrer sur l'apprentissage automatique et sur l'optimisation des différents paramètres pour obtenir le meilleur classifieur (Chandran A. et Christy J. 2015) (Hollstein A., et al. 2016). De même, la classification des sols est impliquée dans un grand nombre d'applications différentes et intéresse beaucoup d'acteurs. Le CESBIO a beaucoup travaillé sur le sujet et a publié beaucoup de résultats (Lassale 2015) (Rodes 2016) et produit pour la première fois une carte de France d'occupation des sols (Figure 3). Déjà présenté précédemment,

[3] https://github.com/BradNeuberg/cloudless

les applications à l'agriculture ont poussé des acteurs américains tels que Descartes Labs (Figure 2) à s'intéresser à ce problème également. Les techniques d'apprentissage sont souvent utilisées dans ce domaine et des informations temporelles sont parfois rajoutées pour améliorer les performances (Rodes 2016).

3.2.4 Les méthodes existantes

Les techniques utilisées pour réaliser ces deux tâches sont principalement issues du traitement d'images et de l'apprentissage automatique. La Figure 17 présente les différentes évolutions du processus de d'apprentissage classées par ordre croissant d'automatisme et ainsi ordre décroissant d'intervention humaine. L'évolution des différents détecteurs de nuages suit également cette évolution.

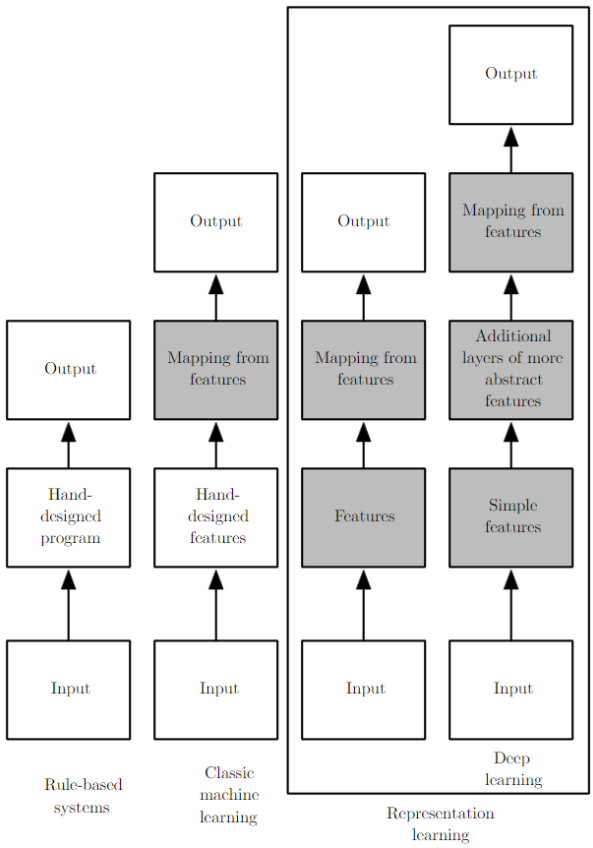

Figure 17: Différents processus d'apprentissage du moins automatique au plus automatique (Goodfellow I., Bengio Y. et Courville A. 2016).

L'objectif est de détecter les nuages à partir des capteurs embarqués sur le satellite : l'entrée de tout système est alors directement la sortie des différents capteurs. Les premiers détecteurs de nuages utilisaient un ensemble de décisions basées sur des indicateurs dédiés crées à partir de la modélisation physique des nuages (Jedlovec 2009). Rapidement, ces détecteurs ont montré des signes de faiblesse en termes de performance et de transférabilité à d'autres satellites car les indices étaient spécifiques aux capteurs embarqués sur le satellite MODIS. Cette méthode a été reprise et appliquée sur des satellites sur

lesquels étaient embarqués des capteurs dédiés à la détection de nuages : les indicateurs étaient alors plus faciles à construire à partir des sorties de ces capteurs. La conception de tels détecteurs nécessite l'installation de capteurs spécifiques et une conception précise des indicateurs. Ces détecteurs ont rapidement évolué vers la seconde famille où une partie des règles de décisions est déduite des données. Le principe est de construire un ensemble d'indicateurs à partir des entrées par défauts et la fonction de décision est déduite de l'analyse de données c'est à dire les images acquises. Les algorithmes d'apprentissage automatique s'occupent d'estimer la fonction de décision (Chandran A. et Christy J. 2015) et la recherche se concentre sur la construction d'indicateurs autrement appelés descripteurs performants. Une partie de ces descripteurs est basée sur différentes opérations de base appliquées aux entrées (Hollstein A., et al. 2016) et l'autre partie utilise des décompositions connues pour créer des descripteurs à base d'ondelettes. Ces descripteurs associés aux méthodes d'apprentissages de l'époque (Chandran A. et Christy J. 2015) ont montré un manque de robustesse par exemple, sur la neige. Les descripteurs ont donc été améliorés grâce à l'étude de la physique des nuages et ses conséquences sur l'image. Par exemple, une méthode développée par le CNES a permis d'extraire l'altitude à partir de l'image satellite et ses métadonnées. Les nuages peuvent être alors détectées précisément (Latry C., Panem C. et Dejean P. 2007) car ils ont une altitude plus élevée que les autres objets de la scène. De la même manière, l'altitude des nuages induit souvent la présence d'une ombre sur l'image. Une détection des ombres sur l'image améliore les performances de détection : l'association d'un nuage et de son ombre permet d'éliminer les fausses détections (Le Hégarat-Mascle S. et André C. 2009). L'information temporelle peut également être rajoutée pour améliorer les performances : l'accessibilité à des séries temporelles sur la même zone permet d'accéder aux statistiques des pixels sans nuages (Hagolle O., et al. 2010) qui, par comparaison avec le pixel d'une image, permet de détecter les nuages. Ces techniques ont considérablement amélioré les performances de détection et l'automatisation de la détection de nuages. Cependant, la plupart d'entre elles nécessitent encore beaucoup d'interventions humaines afin de paramétrer l'extraction des descripteurs mais montrent encore quelques faiblesses en robustesse aux différents paysages. Enfin, la dernière tendance est d'automatiser toute la chaîne en comprenant les descripteurs : cette méthode est récente et encore en développement mais commence à être appliquée à la détection de nuages (CloudLess et (Shi M., et al. 2016)). De la même manière, les méthodes de classification des sols ont suivi la même évolution à la différence que la problématique était suffisamment compliquée pour que des simples règles n'aient pu être créées. Des descripteurs basés sur des indices colorimétriques et de textures ont été utilisés (Huang C., Davis L. et Townshend J. 2002) (Jovanovic V. et Risojevic V. 2014) et récemment, les premières méthodes d'apprentissage profond ont également été testées avec succès sur ce cas d'utilisation (Ienco D., et al. 2017) (Luus F., et al. 2015) (Negrel R., Picard D. et Gosselin P. 2014).

3.3 L'EXTRACTION D'INFORMATION A PARTIR DES IMAGES SATELLITES

3.3.1 Les capteurs

Pour produire des images, les principaux opérateurs tels que Airbus Defence and Space, Thales Alenia Space ou Planets et Digital Gobe pour les américains manœuvrent des constellations de satellites. Une constellation de satellite est un ensemble de satellites qui sont manœuvrés de manière à répondre aux besoins d'acquisition des opérateurs (besoins d'acquisitions, de revisite temporelle, de demandes urgentes). Un satellite inclut différents éléments comme un système de télécommunications pour le

transfert de données, des actionneurs cinématiques pour contrôler son attitude et enfin, la partie centrale concernant les images, un capteur pour l'acquisition. On différencie des familles de satellites suivant la nature des capteurs embarqués.

On peut tout	Name	Low	High
1	Blue	0.45	0.52
2	Green	0.53	0.59
3	Red	0.625	0.695
4	Near Infrared	0.76	0.89
5	Panchromatic	0.45	0.745

Tableau 1: Bandes spectrales acquises par le satellite SPOT 6

Band	Name	Low wavelength	High wavelength	Comments
1	Coastal Aerosol	0.43	0.45	Coastal and aerosol studies
2	Blue	0.45	0.51	Soil, deciduous, coniferous vegetation
3	Green	0.53	0.59	Peak vegetation
4	Red	0.64	0.67	Vegetation slopes
5	Near Infrared	0.85	0.88	Biomass, shorelines
6	Shortwave Infrared 1	1.57	1.65	Moisture
7	Shortwave Infrared 2	2.11	2.29	Moisture
8	Panchromatic	0.5	0.68	Sharper image
9	Cirrus	1.36	1.38	Cirrus cloud
10	TIRS 1	10.6	11.19	Thermal mapping
11	TIRS 2	11.5	12.51	Thermal mapping

Tableau 2: Bandes spectrales acquises par le satellite LANDSAT à titre de comparaison

La deuxième famille est la famille des capteurs actifs qui sont des capteurs basés sur la mesure d'une onde réfléchie par les différents éléments de la scène observée. Une onde est envoyée par le satellite et l'enregistrement du signal rétrodiffusé permet de construire une image (Figure 18). Les satellites radar sont des satellites à visée latérale et monostatique et leurs capteurs permettent d'enregistrer des signaux dans le domaine des hyper fréquences. Les avantages d'un tel capteur sont :

- L'indépendance par rapport au rayonnement solaire, ce qui permet de faire des acquisitions de nuit.
- Le signal traverse les nuages et peut également donner des informations sur une profondeur restreinte du sol
- L'image est cohérente

Les principaux satellites radar sont les satellites ERS, ENVISAT, TerraSar, et COSMO SKYMED qui sont utilisés pour des applications différentes et complémentaires des missions optiques. Malgré ses avantages, les images radar sont entachées d'un bruit multiplicatif et peuvent générer des artefacts tels que les Bright Scatterers.

Il existe encore d'autres familles de capteurs mais ils sont utilisés pour des applications spécifiques et ne sont pas considérés dans l'analyse de cette thèse. Le capteur LIDAR mérite cependant d'être mentionné en guise d'exemple de capteurs exotiques. La télédétection par laser ou LIDAR est une technique de mesure à distance fondée sur l'analyse des propriétés d'un faisceau de lumière renvoyé vers son émetteur. À la différence du radar qui emploie des ondes radio ou du sonar qui utilise des ondes sonores, le lidar utilise de la lumière (du spectre visible, infrarouge ou ultraviolet). Cela permet par exemple de mesurer avec précision la distance entre le satellite et le sol c'est-à-dire estimer l'altitude des objets.

Figure 18: Une image optique et radar de la même zone géographique. Source.

Par la suite, pour tous les capteurs, la finalité est de générer une image qui est en réalité un tableau multidimensionnel (généralement 2 ou 3, et un nombre de bandes qui peut varier suivant le satellite) de valeurs généralement entières. La qualité d'une image est évaluée à partir des différents niveaux de résolutions. La qualité d'une image est le facteur de différenciation entre satellites et opérateurs. Tout d'abord, la résolution spatiale désigne la taille d'un pixel, une case du tableau, au sol. Par exemple, à titre de comparaison, le satellite Pléiades, satellite à très grande résolution, a une résolution spatiale de 30 cm, le satellite SPOT 6, satellite haute résolution, a une résolution de 1.5m et enfin, Sentinel 2 a une résolution de 10m. De même, on parle de résolution spectrale pour comparer les satellites c'est-à-dire le nombre et la largeur de leurs bandes spectrales. A titre d'exemple, le satellite Sentinel 2 possède plus de bandes spectrales que le satellite SPOT 6 (voir le Tableau 1 et le Tableau 2). Une bande spectrale est souvent utilisée pour mesurer une grandeur physique spécifique. Le nombre de bandes limitent donc les applications des images. Le dernier facteur est la revisite temporelle c'est-à-dire la période de temps nécessaire pour prendre une image deux fois dans les mêmes conditions. Un grand nombre d'applications dépendent de l'analyse de série d'images du même lieu ou nécessitent des images extrêmement récentes de lieux spécifiques. Ce facteur est alors essentiel pour ce type d'applications. En particulier, la gestion de satellites en constellation permet de réduire cette période.

3.3.2 Les segments sols

Une fois acquises, les images sont transmises aux segments sols pour traitement et archivage. En effet, l'image est d'abord corrigée (Figure 19) avant archivage et éventuellement traitée afin de calculer les métadonnées. Une fois que l'image est acquise, elle est transmise à une station de réception pour être mise en catalogue avec les métadonnées nécessaires afin d'être accessible par les utilisateurs. En termes d'ordre de grandeur, une image est traitée toutes les 100 minutes par le segment sol de SENTINEL 2A pour produire une image en 1h30. Le segment sol peut représenter une importante charge de travail suivant la fréquence d'acquisition du satellite et du nombre de satellite de la constellation. De manière pratique, le segment sol de Sentinel 2 incluant le stockage des images a été dimensionné pour un flux annuel d'images de l'ordre de plusieurs centaines de Terras voire quelques Peta octets.

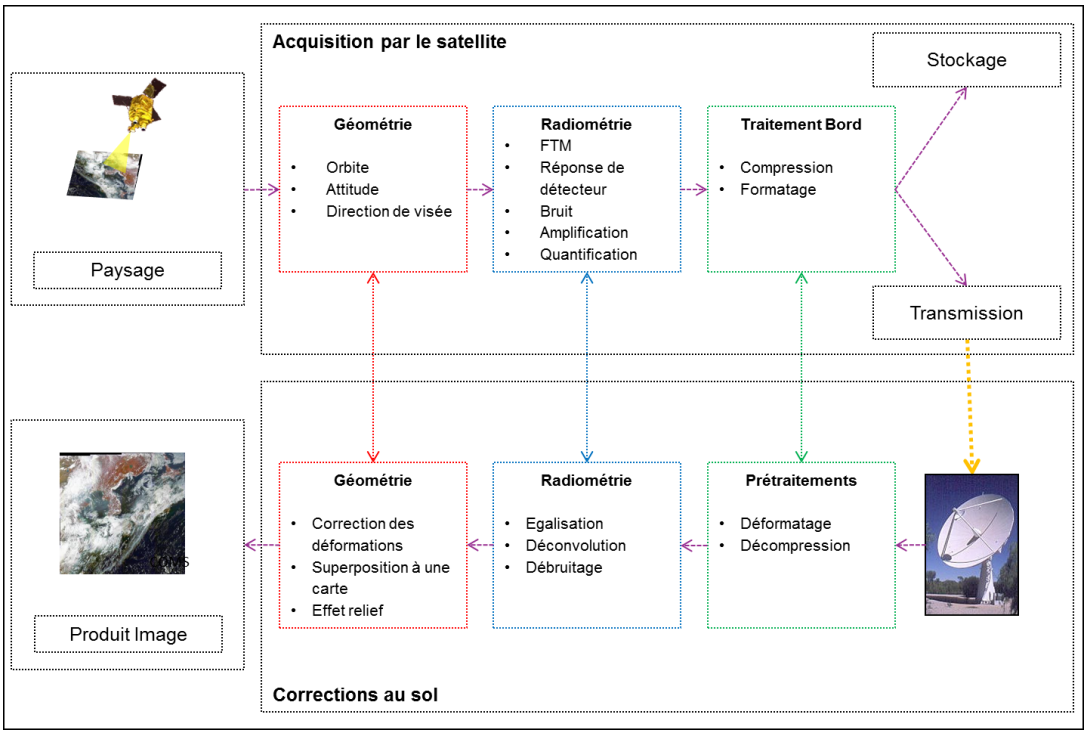

Figure 19: Traitements correctifs du traitement sol et leur sources respectives de perturbation

La prolifération des satellites crée de nouvelles contraintes sur les nouveaux segments sols : ils doivent désormais être robustes au passage à l'échelle c'est-à-dire à l'augmentation du nombre d'images à traiter. L'approche historique consistant à dimensionner des segments sols sur mesure pour un ou deux satellites est désormais dépassée. Les nouveaux segments sols sont amenés à gérer des constellations et la demande en termes de services et de traitements est en croissance. Les traitements vont devoir de plus en plus intégrer des traitements d'extraction d'informations : la détection de nuages et la classification des sols ne sont que des exemples de traitements qui sont amenés à être ajoutés aux chaînes de traitement. L'intervention manuelle majoritairement utilisée pour la détection de nuages par exemple est désormais à proscrire à cause de l'augmentation du flux d'images. Le besoin est recentré sur l'information extraite et la valeur ajoutée des images satellite. Le volume de données à traiter est à la fois un challenge et une opportunité : les nouveaux segments sol doivent traiter plus de données et fournir plus d'informations.

3.3.3 Les différentes familles d'analyse

Plusieurs familles d'analyse peuvent être effectuées sur ces images satellites. La partie suivante a pour but d'introduire les grandes familles de traitements et de placer cette thèse dans son contexte.

3.3.3.1 Détection d'objets

La première famille d'analyse est la détection d'objets dans des images satellites. Créer des modèles permettant de localiser précisément un grand nombre de catégories d'objets reste encore aujourd'hui un

problème non résolu. Dans le domaine de l'image satellite, la détection d'objets est particulièrement utilisée principalement pour le suivi (tracking) mais également pour le comptage, nombre d'occurrences d'objets dans l'image. Le comptage est utilisé afin de créer certains indices : par exemple, le nombre d'avions dans un aéroport ou le nombre de voitures sur un parking. La détection d'objets est utilisée dans beaucoup d'applications comme celles décrites dans la partie 3.1.1. Le but est de pouvoir préciser une liste d'objets présents dans l'image incluant des informations sur la catégorie d'objets et sa localisation précise dans l'image sous forme de cadre. Jusqu'à présent, cela représente une tâche difficile pour diverses raisons incluant la complexité de l'objet à détecter, l'occlusion d'une ou plusieurs parties de l'objet, la localisation précise des objets, et enfin, l'ignorance du nombre total d'instances d'objets dans l'image. De manière pratique, ce type d'analyse nécessite des structures de modèles particulières (Redmon, et al. 2016) (Viola P. et Jones M. 2001).

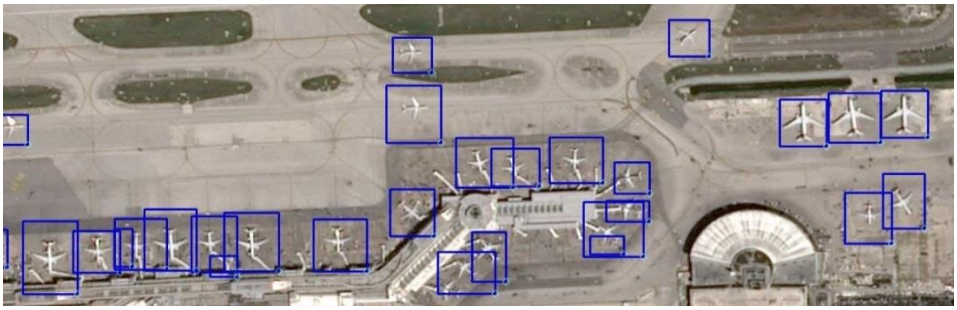

Figure 20: Comptage des avions présents dans un aéroport par image satellite. Source

3.3.3.2 Classification

La classification est une tâche plus simple que la détection en apparence. La classification consiste à attribuer à son entrée une classe d'appartenance. Cette classe est souvent indiquée par un indice. La classification dans le cadre de l'image satellitaire consiste à attribuer un entier indiquant une classe d'appartenance (nuages ou non nuages par exemple, ou encore une classe de terrain) à un pixel. L'application d'un modèle de classification à tous les pixels d'une image permet d'obtenir une carte où tous les pixels possèdent une classe d'appartenance : cette carte est souvent matérialisée sous la forme d'une image où les classes sont indiquées par des couleurs différentes (Figure 21). La classification permet également de faire de la reconnaissance d'objets c'est-à-dire quelle permet de répondre à la question suivante : cette image est-elle l'image d'un avion ? d'un hélicoptère ou d'une voiture ? Le classifieur ne donnera pas d'information sur la localisation de l'objet dans l'image. Notons que les tâches de reconnaissance d'objets et de détection d'objets restent extrêmement liées.

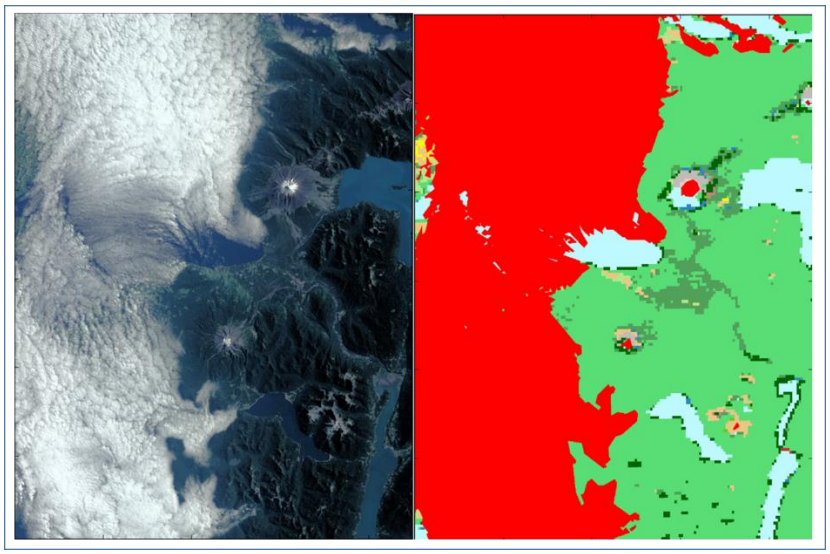

Figure 21: Exemple de classification associé avec une image satellite. Réalisée à partir d'une fenêtre glissante.

3.3.3.3 Segmentation

La segmentation est un autre exemple de traitement appliqué aux images. L'objectif de la segmentation est de diviser l'image en régions homogènes (selon une certaine métrique) et d'attribuer une catégorie à chaque région (Lassale 2015). La segmentation peut être orientée pour réaliser une tâche particulière : par exemple, une segmentation peut volontairement réaliser une opération similaire à la détection de nuages. En effet, si les classes à associer sont spécifiées à l'avance, la segmentation est supervisée et peut réaliser une tâche précise. Mais les classes peuvent être laissées libres et la segmentation est alors non supervisée. La segmentation rassemble au sein d'une même classe d'appartenance des groupes de pixels similaires.

3.3.3.4 Suivi d'évolution

Les images satellites sont souvent utilisées pour faire du suivi d'évolution, c'est-à-dire de l'analyse de séries temporelles et de leurs changements. Principalement, ces analyses consistent à détecter, compter et analyser les occurrences de changements significatifs dans une série temporelle d'images satellites. Beaucoup de travaux ont été réalisés sur le sujet bien qu'il ne soit pas encore résolu. L'explication est dans la complexité du problème : les changements à détecter doivent être significatifs et non crées par la différence de conditions d'acquisition issues du processus d'acquisition (changement d'ombres, etc.). La Figure 22 est un exemple de détection de changement. Beaucoup de changements ont été détecté mais seuls générés par des phénomènes tels que la construction de bâtiments sont intéressants et entourés en rouge. Il existe peu d'exemples disponibles pour ces analyses ce qui rend la tâche d'autant plus complexe.

Figure 22: Détection de changement sur un aéroport en Chine. Divers changements sont détectés mais les changements significatifs issus de construction sont accentués. Source

3.3.3.5 Interaction humaine

Les méthodes d'analyse d'images sont principalement automatiques. Des approches dîtes semi automatiques essaient d'inclure un opérateur humain dans la boucle. En effet, les approches automatiques ont le désavantage de nécessiter de grandes quantités de données expertisées. Ces ensembles de données, constitués par de nombreux échantillons, nécessitent une inspection d'un expert humain et sont donc coûteux à créer. La question se pose alors d'intégrer l'expert dans le processus d'apprentissage de manière à avoir moins de données à classifier. La formation habituelle d'un jeu de données est constituée de l'ajout au hasard d'éléments dans le processus d'apprentissage. L'apprentissage actif (ou Active Learning) consiste à autoriser le modèle d'apprentissage à maîtriser ses données en entrée c'est à dire qu'il peut guider l'expert dans son analyse ou encore poser des questions à l'expert (que penses-tu de cette image ? Est-ce que ma prédiction est correcte ?). L'idée de cette technique est de choisir de manière intelligente les échantillons à classifier par un expert (Chandran A. et Christy J. 2015). Cette technique peut être par exemple utilisée dans le cas de l'analyse d'évènements rares qui ne sont pas pris en compte dans le modèle actuel et qu'il faut incorporer dans le prochain échantillon d'apprentissage.

3.3.3.6 Autres

Il existe bien entendu d'autres applications qui ont beaucoup d'intérêt dans le domaine de l'imagerie spatiale. Un sujet de recherche actuel est, par exemple, l'extraction d'un modèle 3D à partir d'images ce qui permettrait d'avoir des modèles de terrain précis et actualisés plus fréquemment. Les extractions de modèles actuels utilisent des images optiques et radars, quelques fois, couplées avec du LIDAR et pourraient être remplacées par des modèles d'apprentissage automatique. Une autre thématique est l'amélioration d'images : en effet, l'image acquise peut être améliorée en termes de qualité d'image (Figure 23). Un dernier axe de recherche tente d'améliorer l'indexation des images afin de faciliter leur accès : c'est le principe des moteurs de recherche. Il existe encore beaucoup d'applications mais les principales familles ont été présentées.

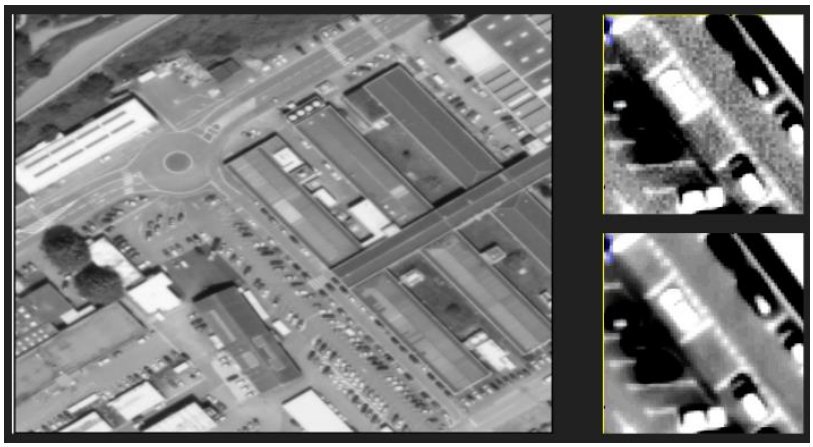

Figure 23: Amélioration d'images panchromatiques. Droite : en haut, image non restaurée, en bas, image après traitement. Source

3.4 OUTILS STATISTIQUES UTILISES DANS LA LITTERATURE

Cette partie est orientée sur les techniques de classification qui comme indiqué précédemment à partir d'une entrée numérique un vecteur ou un tableau renvoient un indice de classe par exemple une classe de couverture, ou un indice binaire de couverture nuageuse. Ces classes peuvent par exemple correspondre à une forêt tropicale, des zones urbaines, de l'eau, des nuages, …. L'image résultante est donc un masque de nuages dans un cas ou une carte d'occupation des sols.

3.4.1 Classification supervisée

La classification supervisée consiste à déterminer une fonction de décision à partir d'un ensemble de données d'apprentissage. Un ensemble d'apprentissage contient des données pour lesquelles la classe est connue. Les ensembles d'apprentissage peuvent être obtenus à partir de connaissances thématiques obtenues auprès d'un expert ou par des bases de données de référence (i.e. carte d'occupation des sols comme celle de la Figure 15). Par exemple, les masques de nuages utilisés ont été détourés sur les images par des opérateurs humains experts en détection de nuages et la base de données de couverture des sols est une carte de référence en libre accès (Broxton P. 2014). Une phase d'apprentissage est ainsi réalisée pour construire une fonction de décision capable de distinguer les classes à partir de l'entrée qui est un tableau ou un vecteur d'attributs. Les méthodes de classification supervisée sont utilisées en télédétection pour la production de cartes d'occupation des sols et la reconnaissance d'objets (Chandran A. et Christy J. 2015) (Hollstein A., et al. 2016) (Lassale 2015) (Rodes 2016). Différentes approches existent pour établir une fonction de décision à partir d'un ensemble de données étiquetées (dont les classes sont connues). Les notations utilisées dans cette partie sont empruntées de (Friedman J., Hastie T. et Tibishirani R. 2001).

Par la suite, nous considérons que S est un ensemble d'apprentissage de N vecteurs de $\mathbb{R}^p$. Chaque donnée est représentée dans $\mathbb{R}^p$ sous la forme d'un vecteur et est notée $\boldsymbol{x}_i$ avec $i \in \{1 \dots N\}$. À chaque donnée est associée une étiquette notée $y_i \in C$ qui identifie la classe de $\boldsymbol{x}_i$. Résoudre le problème de classification se résume à la recherche de la meilleure fonction $f: \mathbb{R}^p \to [0,1]^{card(C)}$ au sens d'une métrique c'est-à-dire une fonction qui renvoie une probabilité d'appartenance à chacune des classes de C. La fonction L est

appelée fonction de coût et permet de mesurer l'erreur entre la prédiction et la véritable classe de $\boldsymbol{x}_i$. La métrique de comparaison appelé risque empirique est la moyenne de la fonction de coût sur le jeu de données. Une méthode de classification consiste à minimiser le risque empirique (Empirical Risk Minimization). On note $\epsilon(S, f)$ le risque empirique de f sur S et se calcule de la manière suivante :

$$\epsilon(S, f) = \frac{1}{N}\sum_{i=1}^{N} L(f(x_i), y_i).$$

La fonction de coût peut prendre plusieurs formes pour s'adapter au problème à résoudre. On représente le plus souvent la classe y_i par son vecteur d'indicatrices $Y = (Y_k), k = 1 \dots K$ où $Y_{y_i} = 1, Y_k = 0\ sinon$ Des exemples courants de fonctions de coût sont présentés ci-dessous avec les fonctions L_2 et l'entropie croisée qui sont principalement utilisées en classification où Y' représente le vecteur de probabilité prédit et Y le vecteur d'indicatrices de la vraie classe:

$$L(\boldsymbol{Y}', \boldsymbol{Y}) = \|\boldsymbol{Y}' - \boldsymbol{Y}\|_2$$

$$L(\boldsymbol{Y}', \boldsymbol{Y}) = -\sum_{k=1}^{K} Y_k \log {Y'}_k$$

Où $K = card(C)$ est le nombre de classes.

On définit une famille $\mathcal{F}$ de fonctions parmi lesquelles chercher la solution. L'objectif est de trouver $f \in \mathcal{F}$ qui minimise le risque ϵ. Le problème d'optimisation peut s'écrire de la manière suivante :

$$f^* = \underset{f \in \mathcal{F}}{\operatorname{argmin}}\, \epsilon(S, f) = \underset{f \in \mathcal{F}}{\operatorname{argmin}}\, \frac{1}{N}\sum_{i=1}^{N} L(f(x_i), y_i)$$

La recherche dans une famille de fonctions est topologiquement complexe et de manière pratique, cette famille est paramétrée de la manière suivante $\mathcal{F} = \{f_\theta \mid \theta \in \Theta\}$. Il est en effet plus facile de chercher un jeu de paramètres θ. Le problème d'optimisation est alors transformé sous la forme :

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmin}}\, \epsilon(S, f_\theta)$$

3.4.1.1 Approche bayésienne

Cette approche utilise un modèle probabiliste où les données sont associées à une variable aléatoire. Notons $\boldsymbol{X}_i$ la variable aléatoire associée à la donnée $\boldsymbol{x}_i$ et Y_i la variable aléatoire associée à y_i. La classification consiste à chercher pour chaque donnée la réalisation Y d'une variable aléatoire représentant la classe. Pour cela, nous cherchons à maximiser la probabilité conditionnelle notée $\mathbb{P}[Y_i = y_i \mid \boldsymbol{X}_i = \boldsymbol{x}_i]$ en utilisant le théorème de Bayes :

$$\mathbb{P}[Y_i = y_i \mid \boldsymbol{X}_i = \boldsymbol{x}_i] = \frac{\mathbb{P}[\boldsymbol{X}_i = \boldsymbol{x}_i \mid Y_i = y_i] * \mathbb{P}[Y_i = y_i]}{\mathbb{P}[\boldsymbol{X}_i = \boldsymbol{x}_i]}$$

- $\mathbb{P}[Y_i = y_i]$ est la probabilité d'apparition des classes y_i

- $\mathbb{P}[\boldsymbol{X}_i = \boldsymbol{x}_i \mid Y_i = y_i]$ est le terme de vraisemblance qui mesure la cohérence entre la réalisation $\boldsymbol{x}_i$ et la classe y_i
- $\mathbb{P}[\boldsymbol{X}_i = \boldsymbol{x}_i]$ est la probabilité à priori d'apparition de $\boldsymbol{x}_i$.

Le terme de vraisemblance est critique pour la classification bayésienne car il contient une grande partie de l'information. Classiquement en traitement d'image et en particulier pour les images satellitaires, des lois gaussiennes sont utilisées pour modéliser les dépendances : $\boldsymbol{X_i}|Y_i = y_i \sim \mathcal{N}(\mu_{y_i}, \sigma_{y_i})$. Les paramètres des lois gaussiennes sont estimés à l'aide de la base d'entrainement.

3.4.1.2 Réseaux de neurones

Cette approche fut initialement inspirée par le fonctionnement du système nerveux biologique. Un réseau de neurones est constitué d'un ensemble de couches connectées. Chaque nœud est connecté avec un ou plusieurs nœuds de la couche précédente et avec un ou plusieurs nœuds de la couche suivante. Un réseau complet de neurones est composé d'une couche d'entrée, d'une ou plusieurs couches cachées et d'une couche de sortie. Chaque nœud dans les couches cachées calcule la combinaison linéaire des valeurs reçues en entrée de la forme $\sum_i w_i s_i$ où w_i est le poids attribué au nœud i de la couche précédente qui émet le signal s_i, puis applique une fonction d'activation non-linéaire pour produire sa sortie. Les fonctions d'activation classiques sont :

- Fonction sigmoïde $x \to \frac{1}{1+e^{-x}}$
- Tangente hyperbolique $x \to \tanh(x)$
- Rectifieur linéaire (Rectified Linear ou ReLu) $x \to \max(0, x)$

Pour le perceptron et la plupart des réseaux de neurones, il y a autant de neurones dans la couche de sortie que de classes. Chaque neurone va donner une probabilité d'appartenance à chaque classe et la classe prédite est attribuée grâce à la probabilité d'appartenance maximale.

3.4.1.3 SVM

Les Séparateurs à Vaste Marge (Support Vector Machines ou SVM) sont des classifieurs binaires qui déterminent l'équation d'un hyperplan optimal qui sépare les données suivant leur classe. Les SVMs ont été grandement utilisés ces dernières années pour le traitement d'images. Afin de simplifier les équations, supposons que $y_i \in \{-1,1\}$. L'équation d'un hyperplan séparateur est de la forme $h(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + \boldsymbol{b} = 0$ pour $w \in \mathbb{R}^p$.

Dans le cas de deux classes, l'hyperplan séparateur optimal recherché par les SVMs est celui qui maximise la marge de séparation entre les données de classe positive et les données de classe négative, c'est-à-dire la distance entre les points les plus proches de l'hyperplan pour chaque classe. Une illustration est présentée dans la Figure 24 qui indique la signification géométrique de la marge m.

Le meilleur hyperplan est obtenu en optimisant la marge de l'hyperplan tout en maintenant la condition suivante qui correspond au fait que la classification reste correcte :

$$\forall i \in [1, N], y_i \, (\boldsymbol{w}^T \boldsymbol{x_i} + \boldsymbol{b}) \geq 1$$

Les données d'apprentissage les plus proches de l'hyperplan sont les vecteurs supports. Lorsque les données ne sont pas séparables linéairement, la solution est de représenter les données d'apprentissage dans un espace de plus grande dimension (éventuellement infinie) dans lequel on cherche un séparateur linéaire. Des fonctions noyaux sont alors utilisées pour représenter une mesure de similarité dans de tels espaces : cette astuce est appelée « kernel trick ». La formulation précédente peut être généralisée à un problème à plusieurs classes avec des approches « one vs one » ou « one vs all ».

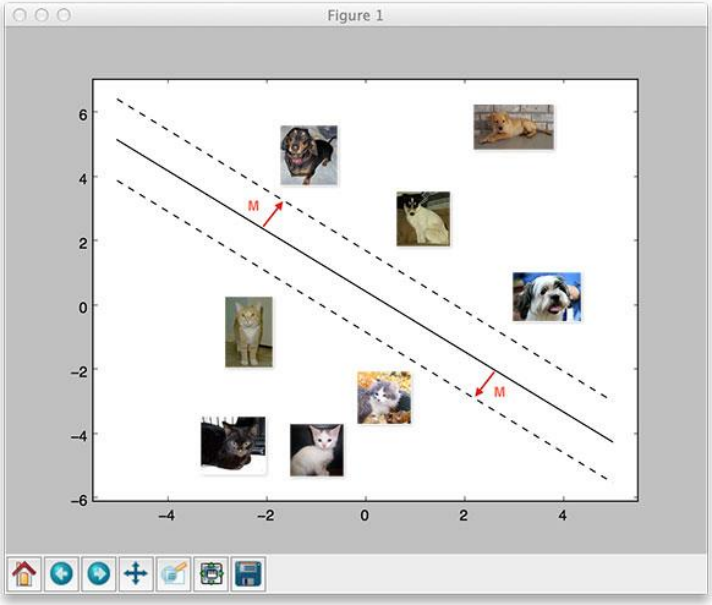

Figure 24: Illustration du principe de la marge. Source

3.4.1.4 Arbres de décisions

Un arbre de décision modélise un processus de classification construit à partir d'une séquence de tests binaires avant de prendre la décision finale. Chaque test est la comparaison de la valeur d'un attribut numérique d'une donnée d'apprentissage avec un seuil. A la suite de la séquence de tests, l'arbre est en mesure d'attribuer une classe au vecteur d'entrée. Un des principaux algorithmes d'arbres de décision est l'algorithme CART ("Classification And Regression Trees") (Breiman L., Friedman J., et al. 1984) qui définit la manière d'apprendre les composantes des tests à partir des données d'apprentissage et quelques techniques pour améliorer les performances en termes de généralisation telles que le « pruning » qui consiste à supprimer les feuilles de l'arbre appris qui ne sont pas statistiquement significatives.

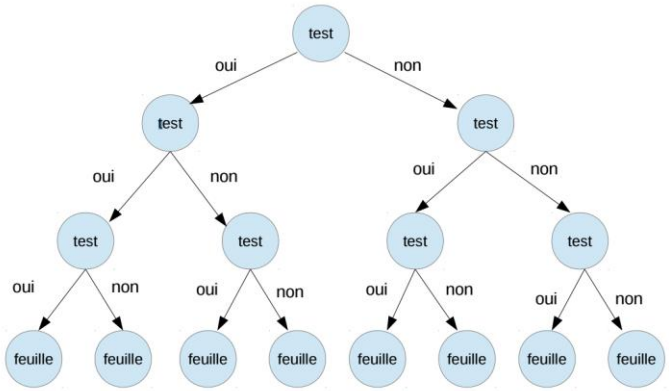

Figure 25: Schéma du principe d'un arbre de décision qui est constitué d'une série de tests.

Chaque test nécessite d'estimer deux paramètres : l'index de l'attribut numérique (une des composantes du vecteur d'entrée) et le seuil de comparaison. Le calcul des composantes du test dans chaque nœud décisionnel est effectué de la façon suivante. Pour chaque attribut, les données en entrée sont triées par valeur. La séquence des valeurs triées est notée $(v_1, v_2, \ldots, v_m)$. Toutes les valeurs de seuil possibles vont être analysées, il y en a $m - 1$ pour m valeurs. Le seuil v_i divise la séquence V en deux nouvelles séquences $S_l = (v_1, \ldots, v_i)$ et $S_r = (v_i + 1, \ldots, v_m)$. Afin d'estimer si le seuil a apporté de l'information, on veut tout d'abord mesurer la pureté d'un nœud (au sens de la classification par exemple). Une mesure classique de pureté est l'entropie de la population de l'ensemble :

$$I(P) = -\sum_{k=1}^{K} \frac{m_k}{m} \log \frac{m_k}{m}$$

Où m_k est le nombre d'éléments appartenant à classe k. Le seuil est alors sélectionné afin d'augmenter au maximum l'information soit :

$$gain(V, S_l, S_r) = I(V) - [I(S_l) + I(S_l)].$$

La mesure d'impureté utilisée dans cet exemple est l'entropie mais d'autres critères permettent de sélectionner le seuil comme le critère de Gini. Les stratégies d'apprentissage d'un arbre de décision telles que celle de l'algorithme CART définissent des conditions d'arrêt d'exploration pour stopper le nombre de tests (la longueur maximale de la séquence) à effectuer avant de faire la prédiction. Ces techniques ont pour effet de réduire le surapprentissage.

3.4.1.5 Random forest

Les arbres de décisions ont montré de bonnes performances pour beaucoup d'applications incluant la détection de nuages (Hollstein A., et al. 2016) et ils sont particulièrement appréciés pour leur interprétation. Cependant, ils ont également montré qu'ils souffraient de sur-apprentissage. En effet, les arbres de décision peinent à maintenir les performances montrées lors de l'apprentissage lors de leur généralisation à de nouvelles images. Afin de pallier ce problème tout en conservant les performances des arbres de décision, dans (Breiman L., Random forests 2001), les auteurs proposent un classifieur, plus

efficace que les arbres de décision, la forêt aléatoire. L'objectif est d'utiliser une technique appelée Bagging qui consiste à générer un ensemble de classifieurs aléatoirement et d'utiliser la moyenne de leur prédiction pour la prédiction finale. Le Bagging est connu pour réduire le risque de sur-apprentissage. Des garanties sur la moyenne de ces classifieurs existent et encouragent l'utilisation de cette technique. En pratique, une forêt aléatoire est composée d'arbres qui ont été appris de manière aléatoire suivant deux stratégies qui sont utilisées à la fois ensemble et de manière séparée suivant les travaux. Chaque arbre est construit à partir d'un sous-ensemble de données d'apprentissage tiré aléatoirement de l'ensemble des données initial (Criminisi A., Shotton J. et Konukoglu E. 2012) (Breiman L., Random forests 2001). Ensuite, pour la construction de chaque seuil, un sous ensemble d'attributs est considéré. Si le sous ensemble est de cardinal 1, on parle de Extremely Randomized Trees (Geurts P., Ernst D. et Wehenkel L. 2006). Les Random Forests ont longtemps été l'état de l'art en termes de méthodes d'apprentissage supervisé et elles sont encore utilisées pour un grand nombre d'applications.

3.4.1.6 Boosting

A l'image des forêts aléatoires, la technique du Boosting (Freund Y. et Schapire R. 1996) a pour but de créer des ensembles d'arbres plus performants qu'un seul arbre. L'idée du Boosting est assez simple : un ensemble est construit itérativement, où à chaque itération, un nouvel arbre est appris à partir des erreurs des précédents et rajouté à l'ensemble. Cette stratégie se focalise sur les exemples difficiles et peut parfois souffrir de sur-apprentissage. De manière à prendre en compte les erreurs des arbres précédents lors de l'apprentissage d'un nouvel arbre, les échantillons de la base de données d'entrainement sont repondérés de manière à donner plus de poids aux exemples difficiles qui ont été mal classés par les précédents arbres. La principale idée derrière le Boosting est de créer un classifieur « fort » à partir d'un ensemble de classifieurs faibles. Un classifieur faible est un classifieur qui est meilleur que le classifieur aléatoire.

3.4.1.7 Autres méthodes

Les principales techniques de classification ont été présentées. Le domaine de recherche en classification supervisée est extrêmement actif et des nouvelles techniques sont conçues à un rythme effréné. Une des techniques connues qui n'a pas été mentionnée auparavant mais qui reste performante est la technique des « k plus proche voisins ». Cette technique repose sur un ensemble d'échantillon et la prédiction d'un nouvel élément est calculée à partir des classes des échantillons les plus proches selon une certaine distance dans l'ensemble de départ. Cette méthode a l'inconvénient de nécessiter un grand ensemble d'échantillons et d'être très sensible à la définition de distance pour calculer les voisins.

3.4.2 Classification non supervisée

3.4.2.1 Besoin

La vérité terrain est difficile à recueillir à cause de son prix et du besoin d'avoir un expert à disposition. Les techniques de classification non supervisée permettent de travailler sur des données sans connaissance à priori de classes d'appartenance. La classification non supervisée consiste à partitionner automatiquement les données par similarité et les classes sont déduites a posteriori. Ce type d'analyse permet de recueillir de l'information sur la structure de la donnée

3.4.2.2 Principales techniques.

Il existe de nombreuses méthodes de classification non supervisée dont les principales sont la classification par l'algorithme « K-Means », la méthode « Mean shift » (Comaniciu D. et Meer P. 2002) et les approches hiérarchiques. Ce sont des méthodes itératives et consistent à déterminer le centre de gravité des données représentant les différentes classes. Pour mesurer la similarité entre les données, ces méthodes utilisent des métriques au sens d'une distance entre les vecteurs d'attributs. Ces approches sont été utilisées pour faire de la segmentation d'image (Lassale 2015) ou pour améliorer les performances de classification.

3.4.3 Détection d'objets

3.4.3.1 Spécificité

La détection d'objets consiste à identifier et localiser dans l'image le ou les objets recherchés. Pour une entrée donnée, la détection renvoie deux informations : la classe d'appartenance tout comme la classification mais également la localisation. Le problème de la détection d'objet est plus complexe que celui de la classification (ou reconnaissance d'objets). Localiser un objet dans une image est complexe et mesurer la performance de la localisation requiert une métrique adaptée. De plus, au contraire de la classification, il peut y avoir plusieurs objets dans la même image et le détecteur doit pouvoir le préciser (Figure 20 et Figure 26). La vérité terrain c'est-à-dire dans le cas présent un ensemble d'images où les objets sont identifiés et localisés par des cadres est complexe à récolter car la récolte demande plus de temps humain par image. Un détecteur d'objet renvoie pour la même entrée les objets présents dans l'image et leur cadre de localisation associé (Figure 26 et Figure 27). Différentes approches existent pour résoudre ce problème. La première approche dérivée des approches de reconnaissance d'objets consiste simplement à prédire une taille de cadre en même temps que la classe d'appartenance (Figure 27). La principale critique faîte à cette méthode est qu'elle ne supporte pas la présence de plusieurs objets dans l'image mais c'est la méthode la moins coûteuse an terme de calcul. La seconde approche sans doute la plus connue est l'approche par proposition de régions ((Ren S., et al. 2015) et Figure 28) où un autre programme extrait un nombre réduit de fenêtres candidates pour contenir un objet et le problème est alors simplifié à un problème de reconnaissance. La deuxième approche plus complexe consiste à renvoyer une carte contenant l'information si un objet se trouve à cet endroit-là dans ce cadre-là.

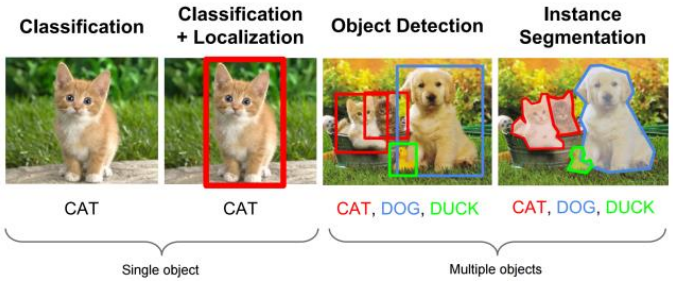

Figure 26: Illustration de la différence entre la classification et la détection d'objets. Source

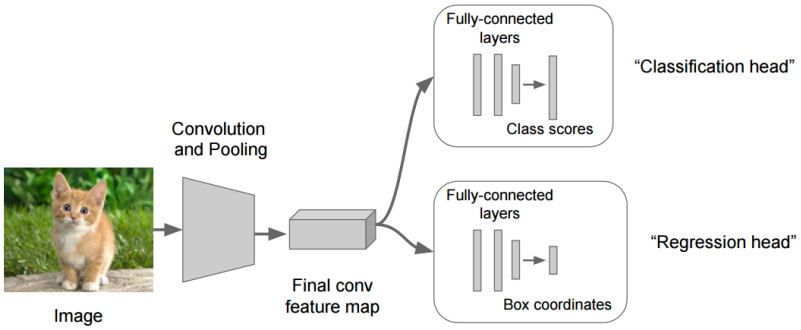

Figure 27: Amélioration de l'approche de classification standard pour faire de la détection d'objets. Une sortie est réservée pour la classification et une autre pour la localisation. Source

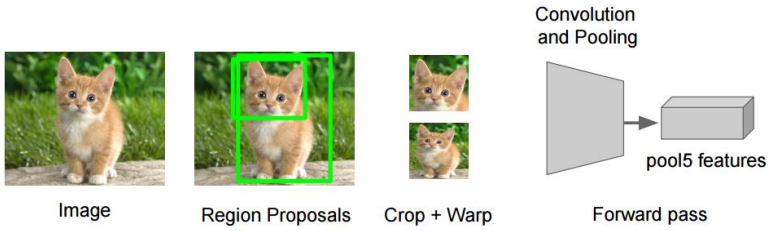

Figure 28: Détection d'objets par proposition de régions Source

3.4.3.2 Détection d'objets à partir d'un classifieur

La reconnaissance et la détection ont toujours été deux tâches extrêmement liées. Plusieurs solutions ont permis de simplifier le problème de détection à un problème de classification qui est plus facile à résoudre. Les méthodes de classification sont mieux maîtrisées que les méthodes de détection d'objets. La question posée est alors suivante : est-ce qu'un problème de détection peut être ramené à un problème de classification ?

Supposons qu'un classifieur pour plusieurs catégories est disponible et que l'on veut construire un détecteur d'objets provenant de ces catégories dans des images. Une des approches serait de recueillir les prédictions d'une fenêtre glissante sur l'image. Les éléments maximisant les scores de classification seront les éléments détectés ; les éléments où le classifieur ne parvient pas à décider seront laissés de côté. L'hypothèse principale sur laquelle repose cette approche est que le classifieur ne pourra décider de la catégorie d'éléments qu'il n'a pas vus durant son entrainement. Or, étant donné que ces éléments n'étaient pas dans son échantillon d'apprentissage, il ne peut y avoir aucune certitude sur les performances du classifieur. Il n'y a aucun contrôle sur la généralisation d'un classifieur.

La deuxième solution consiste à ré apprendre un classifieur avec une nouvelle classe qui sera l'arrière-plan de manière à étendre le domaine de fonctionnement du classifieur, et d'avoir des garanties sur ces performances (Viola P. et Jones M. 2001) (Wang 2014). Ce dernier classifieur permet donc de faire de la détection tout en bénéficiant de la formulation d'un problème de classification. Cependant, introduire l'arrière-plan en tant que catégorie déséquilibre la répartition des populations entre classes. En effet, comparativement au nombre de fenêtres possibles dans l'image, le nombre de fenêtres contenant un objet est très faible, ce qui en fait un problème difficile puisque les statistiques des cas positifs (contenant l'objet) et négatifs sont déséquilibrées. En effet, la classification supervisée est facilitée lorsque la

répartition des populations entre les différents objets est relativement similaire et une mauvaise répartition des classes nécessite l'utilisation de techniques dédiées (Viola P. et Jones M. 2001) (Wang 2014).

3.4.4 Apprentissage profond (Deep learning)

3.4.4.1 Réseaux de neurones profonds

Le but de cette section est d'introduire les principaux concepts derrière l'apprentissage des réseaux de neurones profonds pour finir sur une revue de l'état de l'art des principales structures dans ce domaine. L'explication ci-dessous est concise et reprend des éléments clés introduits dans (Karpathy 2016) qui propose une explication plus avancée et détaillée. Nous avons vu que la clé de tout problème d'apprentissage est de définir la fonction f et les paramètres θ à optimiser pour apprendre et minimiser le risque empirique. En ce qui concerne l'optimisation, la technique couramment utilisée pour cette minimisation avec des réseaux de neurones est la descente de gradient stochastique. En effet, la forme de la fonction le permet et nous donne une formulation explicite de la mise à jour des paramètres (Bottou L. 2012).

$$\epsilon(S,\theta) = \frac{1}{N}\sum_{i=1}^{N} L(f_\theta(x_i), y_i)$$

$$\theta_{t+1} \leftarrow \theta_t + \eta \frac{\partial \epsilon(S,\theta)}{\partial \theta}\Big]_{\theta=\theta_t}$$

Les réseaux de neurones ne sont qu'une façon parmi d'autres de paramétriser la fonction f pour devenir f_θ. Le principe des réseaux de neurones actuels est d'enchainer des opérations élémentaires pour effectuer la classification : les paramètres des fonctions élémentaires seront estimés durant l'entrainement. La règle de composition du gradient permet de calculer le gradient pour n'importe lequel des paramètres et ainsi, la descente de gradient stochastique peut être mise en place. Les différentes opérations définissent ce que l'on appelle un graphe de calcul : le message est transmis dans un sens pour faire la prédiction et dans l'autre sens, pour calculer le gradient et mettre à jour les poids (Figure 29).

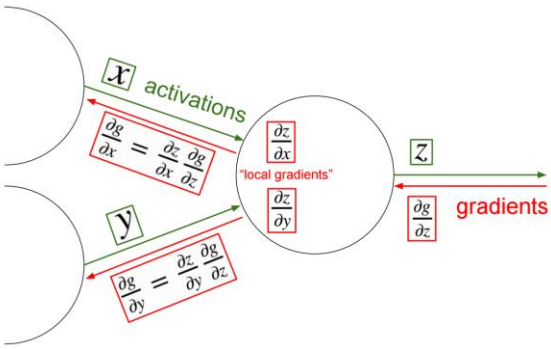

Figure 29: Visualisation de la rétro-propagation. Dans un sens (vert), les activations sont transmises pour la prédiction. Dans l'autre sens (rouge), le gradient est transmis pour la mise à jour des poids. (Karpathy 2016)

Historiquement, les opérations élémentaires évoquées précédemment sont des multiplications matricielles ou des non linéarités (appliquée élément par élément). Un réseau est une composition de plusieurs de ces éléments. Par exemple, la sortie d'un réseau à 2 couches sera $f(\boldsymbol{x}) = W_2\sigma(W_1\boldsymbol{x})$, c'est-à-dire la composition d'une multiplication matricielle par une non linéarité par une nouvelle multiplication matricielle. Notons que σ est une fonction non linéaire appliquée élément par élément qui peut être une sigmoide, une fonction tanh, ou encore un rectifieur linéaire (voir 3.4.1.2). On parle de couche complétement connectée dans ce cas-là (comme illustré sur la Figure 30).

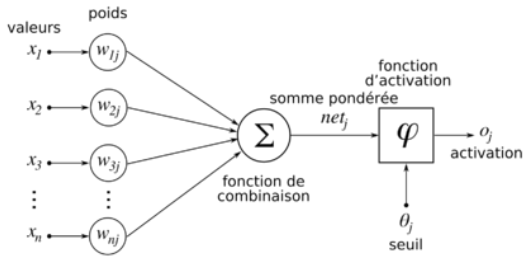

Figure 30: Fonctionnement d'un neurone. (Karpathy 2016)

3.4.4.2 Réseaux convolutionnels

Les réseaux de neurones convolutionnels sont des structures de réseaux de neurones particulières puisque l'opération élémentaire n'est plus une multiplication matricielle mais une convolution. Ces réseaux ont été inventés afin de tirer parti de données possédant une structure (spatiale, temporelle, …) telles que des images, des vidéos, ou encore des signaux temporels. Pour des images, il est courant d'avoir une entrée de taille $256 * 256 * 3$ qui correspond à une image classique sur Internet. Historiquement, des grandeurs ou attributs étaient calculés sur cette image pour servir d'entrées au réseau de neurones. Mais si l'entrée est directement une image soit une liste de presque 200000 entiers, l'utilisation de couches complétement connectées (décrite précédemment) est à proscrire et serait une perte de temps de calcul et de poids à apprendre. Les couches de convolutions permettent de mieux rendre compte de la distribution spatiale grâce à une sensibilité locale tout en maintenant un nombre de poids à estimer raisonnable.

Leur fonctionnement est simple et basé sur une convolution. La sortie est obtenue par convolution de l'image d'entrée et d'un nombre fixé de filtres. La sortie d'un filtre est obtenue en multipliant le contenu d'une fenêtre glissante appliquée à l'image par le filtre (Figure 31).

Une convolution prend en entrée une image I de taille $W \times H \times B$ et un noyau de convolution G de taille $F \times F \times B$ et renvoie le résultat $F = I * G$ (où $*$ symbolise la convolution) défini par :

$$F(i,j) = \sum_{k=1}^{B} \sum_{m=1}^{F} \sum_{n=1}^{F} I(i+m, j+n, k) G(i,j,k)$$

De manière générale, les résultats de plusieurs filtres sont concaténés pour former ce que l'on appelle couramment une carte de features. Dans ce cas-là, le nombre de paramètres à estimer peut se calculer de la manière suivante pour K filtres :

Paramètres par filtres : $F \times F \times D$, pour un total $F \times F \times D \times K$ poids et K biais.

Des couches de pooling ou de sous échantillonnage sont également ajoutées aux opérations élémentaires. Elles permettent de décroitre la taille de la représentation avec un sous échantillonnage spatial : de cette manière, elles permettent de contrôler le sur-apprentissage. Le sous échantillonnage est appliqué de manière indépendante sur chaque carte d'activation (sortie correspondant à un filtre). Par exemple, dans une configuration classique, l'utilisation d'un sous échantillonnage par maximum (Max Pooling) dans un voisinage de 2×2 avec un pas de 2 permet de réduire la taille de l'entrée par 4 (par 2 dans chacune des dimensions).

Supposons que l'entrée est une matrice I de taille $H \times W$, le voisinage du pooling F et le pas s alors le résultat du max pooling F est défini comme suit :

$$F(i,j) = \max\left(\{I(i * s + m, j * s + n)\}_{m=1\dots F, n=1\dots F}\right).$$

L'opération consiste simplement à calculer le maximum d'une fenêtre glissante de pas s. Le max pooling est appliqué de manière indépendante sur chaque bande de l'image si l'image possède plusieurs bandes et les résultats sont concaténés pour former la sortie.

L'enchainement de couches convolutionnelles, de couches de pooling et enfin de couches connectées est souvent désigné par le terme d'architecture de type ConvNet (LeCun Y., et al. 1998). Une architecture typique suit le schéma suivant :

$$[\text{INPUT}, (\text{CONV}, \text{POOL})^k, (\text{F C})^l]$$

$k = 2, l = 2$, [INPUT, CONV, POOL, CONV, POOL, FC, FC] (LeCun Y., et al. 1998)

De la même manière que pour les réseaux de neurones, la dernière couche connectée a autant de sorties que de classes à prédire et est utilisée pour faire la classification.

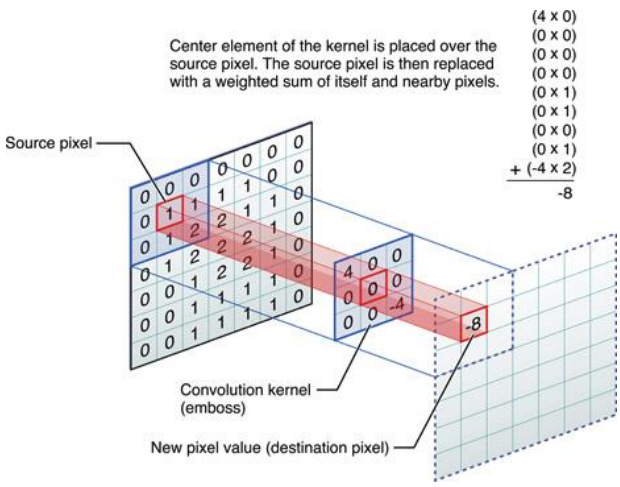

Figure 31: Fonctionnement d'une convolution sur une image. Source

3.4.4.3 Apprentissage par transfert (Transfer learning)

L'apprentissage par transfert désigne un ensemble de techniques permettant de transférer des connaissances d'une ou plusieurs tâches initiales vers une nouvelle tâche. Il est souvent appliqué pour transférer la reconnaissance d'un type d'objets vers un autre. Par exemple, un classifieur dont les catégories eau, ville, forêt, … seraient similaires à un autre cas d'étude pourrait servir pour apprendre un nouveau classifieur pour ce nouveau cas d'étude sans ré apprendre les poids d'un réseau entier. Cet ensemble de techniques est souvent utilisé pour pallier le manque de données, la durée d'apprentissage, le manque de puissance de calcul, et le manque de connaissances en apprentissage profond.

Le cas des réseaux de neurones a beaucoup été étudié (Donahue J., et al. 2014) pour faire du transfert. Il existe plusieurs approches pour le transfert de réseaux de neurones : soit une partie d'un réseau existant est réestimée pour la tâche cible (Figure 32) soit une des couches est sélectionnée afin de servir d'entrée à un nouveau classifieur. Ce mode de fonctionnement est facilité avec la mise en disposition en accès libre d'un grand nombre de réseaux déjà appris pour certaines tâches telles que la reconnaissance d'images, de monuments historiques, … La donnée est une ressource rare et stratégique et afin d'en maximiser son utilisation, une technique développée avec l'aide de l'apprentissage par transfert consiste à utiliser une grande base de données moins coûteuse, mais moins précise pour effectuer un premier apprentissage puis améliorer le classifieur ainsi obtenu pour la tâche cible sur une plus petite base de données mais beaucoup plus précise (Figure 32).

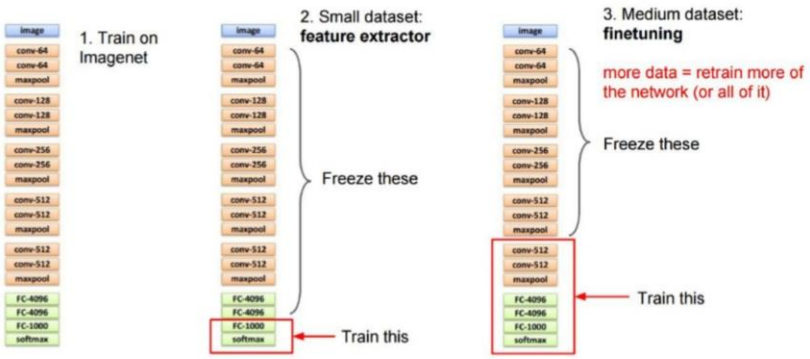

Figure 32: Principe du transfer learning avec le réseau VGG16. Source

3.4.4.4 Grandes architectures

Les réseaux de neurones profonds sont des algorithmes puissants et sont devenues depuis quelques années extrêmement populaires. Une grande partie de leur succès repose sur la conception minutieuse de l'architecture du réseau et en particulier, sur la conception de bloc élémentaire. L'objectif de cette section est de retracer l'histoire de la conception des réseaux de neurones au cours de ces dernières années afin de mettre en évidence les différents blocs inventés et utilisés dans les réseaux actuels. D'autres travaux sont néanmoins à indiquer (Canziani A., Paszke A. et Culurciello E. 2016) pour une analyse plus approfondie et une comparaison de tous les réseaux répertoriés ici. La plupart des réseaux présentés sont des réseaux utilisés pour faire de la classification c'est à dire de la reconnaissance d'images et les réseaux développés pour la segmentation (Ronneberg O., Fischer P. et Brox T. 2015) ou la détection d'objets (Redmon, et al. 2016) ne sont pas considérés dans cette partie.

Les premiers réseaux profonds à avoir détrôné les approches classiques pour la classification sont les réseaux de la famille des ConvNets. Le réseau du nom de LeNet issu des travaux d'un des pères fondateurs de l'apprentissage profond Y. LeCun (LeCun Y., et al. 1998) est le premier réseau à avoir prouvé la faisabilité de l'apprentissage profond par l'expérimentation : à l'époque, le réseau était constitué de 2 couches de convolutions et de 2 couches entièrement connectées pour faire de la reconnaissance de chiffres (LeCun Y., et al. 1998).

Grâce à l'amélioration des conditions d'apprentissages en termes d'optimisation avec en particulier l'ajout de moments dans la descente de gradient (Kingma D. et Ba J. 2015) et de moyens de calcul tels que l'accès aux GPUs et/ou au cloud, le nombre de paramètres et la profondeur des réseaux ont augmenté. Les premiers rectifieurs linéaires (ReLu) ont été utilisés afin de contrer l'effet de la dilution de gradient (vanishing gradient) qui ralentissait l'optimisation également. La seconde génération de réseaux profonds dont font partie AlexNet (Krizhevsky A., Sutskever I. et Hinton G. 2012) et Overfeat (Sermanet P., et al. 2014) sont donc plus profonds et larges que leur ancêtre LeNet mais respectent la structure des ConvNets.

La génération suivante est composée en majeure partie des réseaux de la famille des VGG (Simonyan K. et Zisserman A. 2015) qui utilisent les convolutions 3x3 qui étaient évitées jusque-là. Des recherches ont montré que leur enchainement permettait de reconstruire les convolutions 11x11 d'AlexNet tout en contrôlant le nombre de poids à estimer. Les convolutions 3x3 sont encore préférées de nos jours pour la conception de structures. L'objectif à l'époque était de trouver un ensemble de règles de design de structure de manière à contrôler le nombre de poids à estimer par réseau (Iandola F., et al. 2016). Cependant, il reste beaucoup de paramètres à apprendre en particulier pour les réseaux de la famille des VGG (VGG-16, Figure 36 et VGG-19) : ces réseaux sont connus pour être flexibles (conviennent pour différentes tâches) mais extrêmement lents à la fois pour l'apprentissage et la prédiction.

L'approche du Network-in-network (NiN, (Lin M., Qjang C. et Schuicheng Y. 2013)) a introduit l'utilisation des convolutions 1x1 de manière à mélanger les différentes activations et augmenter le pouvoir représentatif du réseau. L'architecture NiN utilisent des couches entièrement connectées (MLP) spatiales après chaque convolution, afin de mieux combiner les caractéristiques avant une autre couche. Cette idée sera reprise plus tard dans les architectures les plus récentes comme ResNet et Inception.

La performance des réseaux est désormais démontrée grâce aux différentes compétitions remportées par des réseaux profonds mais jusqu'en 2014, le principal défaut de l'apprentissage profond restait la lenteur des réseaux lors de la prédiction et l'apprentissage. A l'époque, les équipes de Google avaient pour but de développer des réseaux plus rapides en termes d'opérations avec moins de poids à estimer tout en gardant les mêmes performances (Figure 37). C'est ainsi que vit le jour la structure Inception (Szegedy C., et al. 2015) avec un module illustré sur la Figure 33 qui tire parti des conclusions historiques présentées précédemment. Les analyses (Szegedy C., et al. 2015) ont montré que les réseaux de type Inception ont largement amélioré le temps de prédiction et l'utilisation des paramètres du réseau. Un facteur 10 en termes d'opérations peut être observé entre une architecture du type VGG et Inception pour des performances similaires. Plusieurs versions du réseau Inception sont apparues courant 2015 incorporant chacune de nouvelles modifications. Par exemple, la normalisation de batch ou Batch Normalisation consistant à normaliser les cartes d'activations a amélioré la convergence et la régularisation des réseaux : chaque couche a désormais en entrée des données blanchisées de la même manière.

En décembre 2015, les premiers éléments de la famille ResNet (He K., et al. 2016) apparaissent en introduisant une idée simple : le principe des blocs résiduels (Figure 34 et Figure 35). Deux couches de

convolutions peuvent se suivre directement et la sortie peut être directement connectée à l'entrée (Figure 34). Cette idée assez simple en apparence a eu de grandes conséquences et en particulier, ce fut la première fois qu'un réseau dépassant une profondeur de 100 couches fut entrainé. Cette nouvelle structure permet à l'information d'éviter certaines convolutions et ainsi, la prédiction (Szegedy, et al. 2016) provient seulement de l'activation d'un sous ensemble des couches. Actuellement, les structures continuent d'évoluer : les ResNets ont été intégrés dans Inception pour donner la quatrième version d'Inception et un nouveau type de convolution a été rajouté pour dériver Xception de Inception V4 (Chollet 2016). L'état actuel de la recherche de meilleures structures montre que les structures ne sont pas stables et sont amenées à continuer d'évoluer.

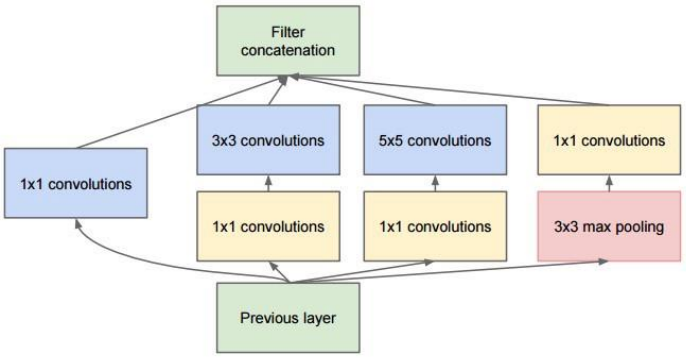

Figure 33: Module Inception utilisé dans la famille des réseaux Inception. (Szegedy C., et al. 2015)

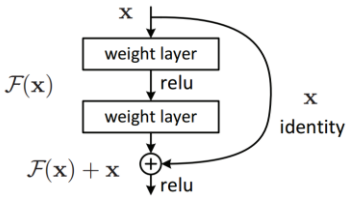

Figure 34: Mapping résiduel utilisé dans les ResNets. (He K., et al. 2016)

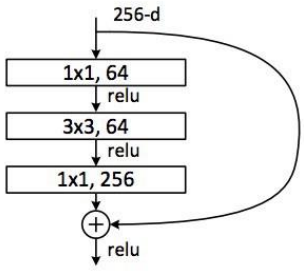

Figure 35: Implémentation d'un block résiduel. (He K., et al. 2016)

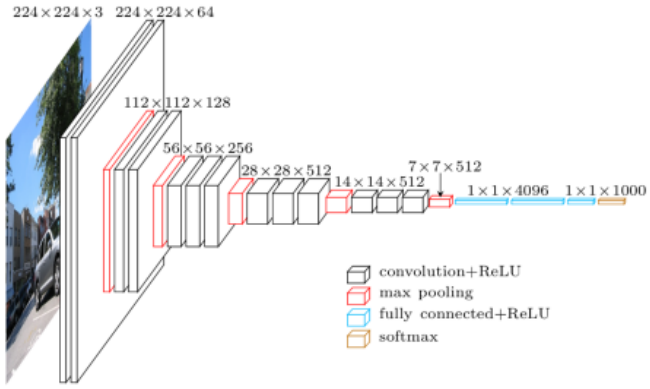

Figure 36: Structure du réseau VGG16. Source

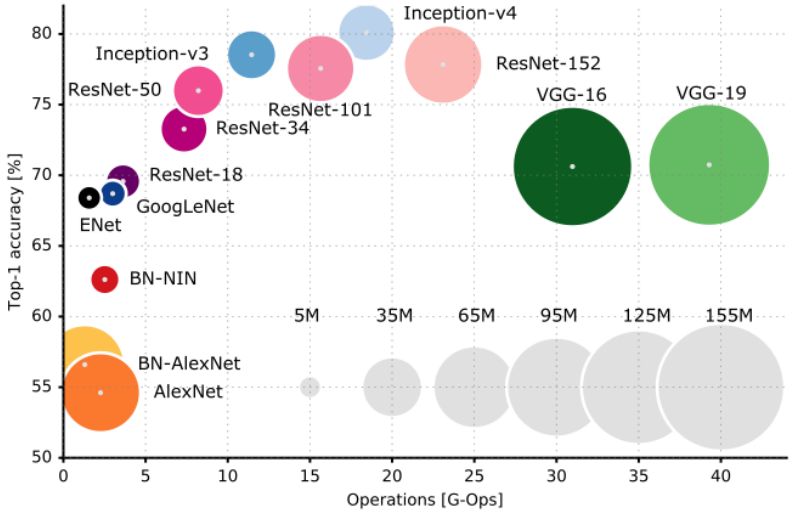

Figure 37: Comparaison des différentes structures de réseaux convolutifs pour la classification. Performance/Nombre d'opérations nécessaires pour faire la prédiction. La taille des cercles indique le nombre de poids à apprendre. Source. Source

3.4.4.5 Réseaux génératifs

A la différence des réseaux précédents de classification, le but de cette famille de réseaux est la génération d'échantillons. Contrairement aux réseaux convolutionnels, ces réseaux acceptent comme entrée un tableau ou un vecteur de flottants et renvoient une image en 2 dimensions (par exemple, 256x256x3) (voir Figure 38) dans les cas où les échantillons sont des images. Cette opération semble plus que complexe que la classification car cette fois ci, l'information est étendue à la place d'être compressée : la taille en sortie est supérieure à la taille en entrée. Les réseaux utilisés ont un nombre de paramètres inférieur au nombre d'images de la base de données d'entrainement, ainsi les modèles compressent l'information afin d'apprendre la structure minimale de la donnée. A l'heure actuelle, les réseaux génératifs actuels sont avant tout utilisés pour la reconstruction d'image, le dé-bruitage et l'amélioration d'images (super résolution). Apprendre à générer des images d'une base de données d'images naturelles est équivalent à modéliser à quoi le monde réel ressemble.

L'objectif des réseaux génératifs est de générer des images provenant de la même distribution que la base d'entrainement. De manière pratique, le réseau génère une image aléatoire de la base de données à partir d'un vecteur ou tableau aléatoire. L'approche Generative Adversarial Network (GAN) (Goodfellow 2016) utilisée pour apprendre de tels réseaux est d'introduire un second réseau qui essaye de faire la distinction entre les images générées par le premier réseau et les images issues de la base de données. Les deux réseaux sont optimisés tour à tour de manière à rendre la tâche plus difficile au réseau opposé. Sur le même principe que les réseaux de classification, la rétro-propagation est utilisée pour estimer les paramètres des réseaux durant la phase d'apprentissage.

Figure 38: Droite: Images utilisées pour l'apprentissage du réseau génératif. Gauche: Images générées par le réseau. Source

3.4.4.6 Optimisation

L'apprentissage profond est un domaine expérimental où l'expérimentation fait office de preuve. Comme il a déjà été évoqué auparavant, le développement de l'apprentissage profond est lié à l'évolution des moyens de calcul et aux évolutions des techniques d'optimisation. En particulier, déjà connu dans les années 80, la descente de gradient stochastique (Bottou L. 2012) connaît actuellement un grand succès et est principalement utilisée pour optimiser ces réseaux. Les défauts de la descente de gradient stochastique sont connus et des corrections numériques tels que les moments ont permis de pallier certains effets. Le point central de l'optimisation des réseaux profonds est de pouvoir calculer le gradient par rapport à chaque variable à estimer. Cela est réalisé par le biais de la « chain rule » détaillée dans (Karpathy 2016) qui permet de calculer facilement le gradient d'une composition de fonctions. Par chance, les réseaux profonds sont constitués d'un enchainement d'opérations élémentaires (convolutions, sous échantillonnage, couches entièrement connectées) dont l'agencement varie suivant les réseaux. Les gradients des blocs élémentaires sont connus et par composition, le gradient des poids à estimer est calculé. Le développement de la différenciation automatique a permis d'automatiser ce processus et à la communauté de se concentrer sur d'autres problématiques de l'apprentissage profond que le calcul des gradients.

La régularisation est également une composante importante de tout problème d'apprentissage supervisé (Figure 37). La régularisation est un processus courant en apprentissage consistant à ajouter de l'information sous forme de contraintes à un problème pour éviter le surapprentissage et améliorer les performances en généralisation. Cette information prend généralement la forme d'un terme supplémentaire pénalisant la complexité du modèle. Par exemple, un terme de pénalité sur la norme des poids du réseau est souvent ajouté au problème d'optimisation pour limiter la complexité. Dans le cas de

réseaux neuronaux, on peut également ajouter des couches de dropout qui vont éteindre aléatoirement une partie des nœuds de la couche courante pour éviter une spécialisation du réseau global (Srivastava N., et al. 2014). Le dropout est beaucoup utilisé et a déjà montré à plusieurs reprises sa capacité à améliorer les performances en généralisation (Srivastava N., et al. 2014). Le dropout nécessite la définition de la proportion de neurones qui sont éteints aléatoirement à chaque itération. Les termes de régularisation sont considérés comme des hyperparamètres à ajuster lors de l'optimisation et ont pour principale utilité d'améliorer la convergence (Krizhevsky A., Sutskever I. et Hinton G. 2012) de l'algorithme de mise à jour des poids du réseau.

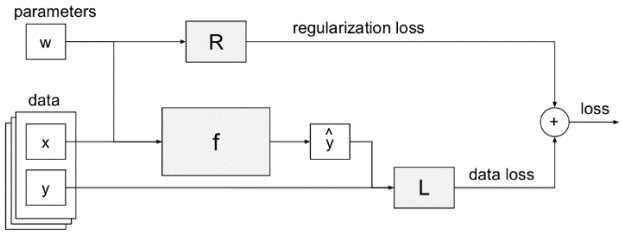

Figure 39: Composantes essentielles d'un problème d'apprentissage automatique (Karpathy 2016).

A part la structure, les seuls facteurs nécessaires pour définir complétement un problème d'apprentissage sont la méthode d'optimisation et la fonction de perte (Figure 39). La méthode d'optimisation est critique à définir et comporte plusieurs composantes à choisir.

- Tout d'abord, un grand nombre de modèles de descente de gradient stochastique existent et se différencient la plupart sur la formulation de leur moment c'est-à-dire la formule de mise à jour des poids.
- Le choix du taux d'apprentissage (Learning Rate) est plus critique (Bottou L. 2012) et en particulier, le choix de son évolution temporelle (fixe, décroissance linéaire, …).
- La taille du batch utilisé à chaque itération a également un impact important sur la vitesse de convergence, le temps de calcul et sur les performances finales (Goyal P., et al. 2017): un compromis entre rapidité et performance reste encore à définir.

En général, tous les hyperparamètres d'optimisation sont choisis par validation croisée (Bergstra J. et Bengio Y. 2012).

Le dernier élément crucial à déterminer est la fonction à minimiser. La fonction à minimiser est la fonction de perte utilisée pour comparer deux opérateurs (humains ou automatiques). Un catalogue de fonctions de perte classiques pour la classification est disponible et inclut, pour les plus connues, la Mean Squared Error et l'entropie croisée (cf 3.4.1). La flexibilité des réseaux de neurones profonds et la différenciation automatique encourage la conception de fonction de perte plus complexe telles que l'aire sous la courbe ROC par exemple, ou des fonctions de perte asymétriques.

En conclusion, le choix parmi toutes les combinaisons possibles est principalement fait de manière empirique où les performances sont évaluées a posteriori et sont spécifiques à un cas d'utilisation. L'aspect empirique de l'apprentissage profond explique l'absence d'une méthode commune globale.

3.4.4.7 Utilisation actuelle

Un problème d'apprentissage automatique est en réalité un ensemble de composants (couches, optimiseurs, augmentation de données, régulariseurs) qui peuvent être assemblées et mélangées ensemble afin de créer un réseau et son utilitaire d'apprentissage (Figure 39). Des structures classiques de réseaux de neurones existent et ont montré de très bonnes performances pour leur application (cf 3.4.4.4) : cependant, ces structures « classiques » nécessitent d'être adaptées pour un nouveau cas d'utilisation. Les hyperparamètres doivent être optimisés pour chaque nouvelle application et sont de manière générale choisis par recherche aléatoire (Bergstra J. et Bengio Y. 2012) et validation croisée. L'apprentissage profond tire profit des moyens de calcul (cloud et GPUs) à disposition en lançant en parallèle des apprentissages sur un grand nombre de configuration (structure de réseaux et utilitaire d'apprentissage). La force de l'apprentissage est sa flexibilité et son automatisation. La formulation stricte d'un problème sans son optimisation se résume à la définition de deux éléments : la fonction de perte à optimiser et la base de données d'apprentissage. Les paramètres de structures et d'optimisation peuvent être choisis par validation croisée moyennant une puissance de calcul adéquate. La conjonction des éléments suivants la différenciation automatique, l'enchainement des couches élémentaires et la puissance de calcul à disposition est le principal facteur expliquant la réussite actuelle de l'apprentissage profond.

3.4.5 Différence entre le traitement des images classique et des images satellites

La plupart des réseaux présentés précédemment ont remporté des compétitions basées sur des bases de données d'images classiques provenant du Web (Deng J., et al. 2009) mais les images satellites ont des propriétés bien différentes. La différence structurelle entre les deux types d'images soulève la question de la transmission de la performance d'un cas d'utilisation à l'autre. L'objectif de l'analyse d'images satellites est majoritairement de détecter des objets ou des types de terrains et les analyses suivantes se concentrent sur ces deux applications.

3.4.5.1 Echelle

Les échelles des objets à détecter comparativement à la taille de l'image sont souvent très petites. En effet, l'analyse d'images satellites peut servir à détecter des véhicules (avions cf Figure 20, voitures, …) sur des images satellites qui ont une taille de plus de 50 kilomètres de large. Le nombre de pixels représentant l'avion est négligeable par rapport au reste : quelques pixels comparativement au total d'un million de pixels de l'image. De la même façon, les images satellites ne sont pas centrées sur un ou plusieurs objets d'intérêt contrairement aux images classiques (qui sont souvent des photographies). Les images satellites offrent la possibilité d'observer une grande zone dans laquelle la taille, le nombre et la nature des objets varient. Cette utilisation est liée au fait qu'une image satellite est complexe à acquérir et par conséquence, chère : son utilisation est maximisée afin de détecter un maximum d'objets.

Lorsque les instances à détecter sont liées à des phénomènes de couvertures, le second problème d'échelle est la variabilité d'échelle des objets à détecter. Cela est particulièrement vrai pour la détection de nuages mais également pour les couverts, pour lesquels une ville n'a pas une taille fixe et peut varier du simple, au double, au quintuple. Les nuages, quant à eux, sont connus pour varier en taille : les nuages de différentes tailles sont souvent généralement issus de familles différentes.

3.4.5.2 Invariance par rotation

Le processus d'acquisition d'une image satellite induit également une invariance par rotation. En effet, la prise de vue est réalisée par le haut et par conséquence, un avion ou un véhicule ou un champ n'a pas plus de chances d'apparaître selon une certaine orientation dans une image satellite. De la même manière, les nuages n'ont également pas d'orientation particulière. En revanche, dans les images classiques, la gravité induit une certaine orientation des objets : les personnes se tiennent debout comme les animaux et les objets ne flottent pas. De manière générale, la rotation d'une image satellite n'affecte pas la classification mais la rotation d'une image classique n'est pas naturelle et affecte la classification.

3.4.5.3 Taille des images

La taille des images a également un autre impact que celui sur les échelles des objets à détecter. En effet, la taille des images satellite est plus grande que la taille d'images classiques : le facteur de taille varie entre 10 et 100 suivant les satellites. De manière générale, on peut considérer qu'une image classique possède une taille de l'ordre de quelques centaines de pixels par centaines de pixels : une image 100000x100000 pixels est une image en pleine résolution. La conséquence principale est qu'une telle image peut tout à fait contenir dans la mémoire vive (RAM) de l'ordinateur et la prédiction ou la segmentation peut être faîte sur l'image totale. Cependant, les images satellites ont des tailles de l'ordre du millier ou dizaine de milliers de pixels. Une image Sentinel 2 aura une largeur de 100.000 pixels tout comme Pléiades ou SPOT 6. Ces images ne contiennent pas en RAM et doivent être traitées par tuile. Les traitements tels que la classification et la segmentation seront réalisées par tuile et rassemblés pour obtenir le résultat final. Une attention particulière est consacrée à cet aspect puisque les calculs d'un réseau de neurones profond sont intenses. Malgré le grand nombre de pixels, la plupart d'entre eux sont extrêmement corrélés : c'est-à-dire qu'il y a une homogénéité spatiale et ils ne représentent pas une grande diversité par image (de terrains par exemple). La diversité est souvent acquise avec le nombre d'images à traiter plus que le nombre de pixels.

3.4.5.4 Structure

Le cas des avions dans les images satellites se rapproche beaucoup de la détection d'objets dans des images classiques pour plusieurs raisons : la principale est la recherche d'une structure spécifique dans les images. En effet, un avion est longiligne et possède deux ailes à l'image d'un visage qui a une forme ronde et possède des yeux et un nez. L'apprentissage profond a pour but de combiner des éléments locaux de manière à créer des concepts généraux : il extrait donc la structure de l'objet (comme l'a montré l'analyse des réseaux pour la détection de nuages Figure 40). Malheureusement, le cas des avions n'est qu'un cas particulier dans les images satellites et les objets ou phénomènes à détecter ont beaucoup moins de structure. En effet, les nuages, les forêts, les villes, les nappes d'hydrocarbures n'ont par exemple pas vraiment de structure et leur détection est basée sur l'analyse des textures, des couleurs et des formes. La structure d'un réseau de neurones profond l'encourage à chercher une composition d'éléments caractéristiques : ce qui explique les contre-exemples les plus importants (voir Figure 41). L'utilisation des réseaux de neurones profonds pour les images satellites soulève la question : est-ce que la transposition des réseaux actuels est suffisante ? Ou faut-il une conception spécifique à l'image satellite ?

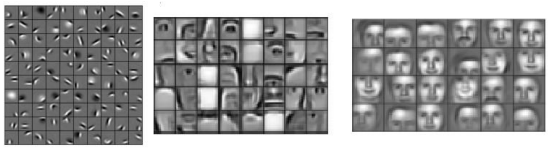

Figure 40: Features hiérarchiques apprises par le réseau de neurones. Source

Figure 41: Erreurs commises par le classifieur appris sur Imagenet entre un chihuahua et des muffins. Source

3.5 LES OUTILS INFORMATIQUES

L'arrivée de nouveaux moyens de calcul sur le marché public a grandement contribué au développement de l'apprentissage automatique, en particulier de l'apprentissage profond. Les outils informatiques ont désormais une place essentielle dans ce domaine et quelques-uns sont présentés dans la section suivante. Les GPUs ne sont pas présentés malgré le fait qu'ils aient également contribué au développement car c'est avant tout la location d'outils informatiques incluant des CPUs, des GPUs, et de l'espace de stockage qui révolutionne actuellement le monde.

3.5.1 Le cloud public

Le cloud computing ou l'informatique en nuage est l'exploitation de la puissance de calcul ou de stockage de serveurs informatiques distants par l'intermédiaire d'un réseau, généralement internet. Les serveurs sont loués à la demande, le plus souvent par tranche d'utilisation selon des critères techniques (puissance, bande passante, etc.) mais également au forfait. Le principal avantage du cloud computing est sa grande souplesse : il est possible de gérer son/ses serveur(s) et son utilisation s'adapte aux besoins de l'utilisateur. Le cloud computing bouleverse l'utilisation historique des systèmes informatiques où des serveurs de calcul étaient mis à disposition au sein d'une même entreprise. Le cloud computing est un basculement de tendance : au lieu d'obtenir de la puissance de calcul par acquisition de matériel et de logiciel, le consommateur se sert de puissance mise à sa disposition par un fournisseur via internet.

Les caractéristiques essentielles du cloud sont :

- La disponibilité mondiale en libre-service : quelle que soit la demande, des machines seront disponibles et prêtes à être louées.
- L'élasticité : le nombre de machines louées s'adapte au besoin de l'utilisateur.
- L'ouverture : Un protocole de communication uniformisé permet un accès fluide depuis n'importe quel endroit.
- La mutualisation : Les ressources informatiques du parc du fournisseur sont rationalisées. Une machine non utilisée est louée à un autre utilisateur. L'attribution de ressources (matériel, logiciel, trafic réseau) est automatique.
- Le paiement à l'usage : La quantité exacte de service consommée dans le cloud est mesurée, à des fins de contrôle, d'adaptation des moyens techniques et de facturation

Un cloud peut être public, privé ou communautaire. Un nuage public est mis à disposition du grand public. Les services sont généralement mis à disposition par une entreprise qui manipule une grande infrastructure lui appartenant par exemple, Google, Amazon, ou Microsoft. Un nuage privé est destiné exclusivement à une organisation.

Ce type de technologies est nouvelle et révolutionne divers domaines incluant celui de l'intelligence artificielle. La rentabilité du cloud est supérieure à celle de l'achat d'une machine de calcul ; de même, l'accès aux GPUs, facteur essentiel de succès dans le monde du Deep Learning, a été grandement simplifié grâce à leur location.

Les fournisseurs de cloud investissent de plus en plus dans le machine et le Deep Learning de manière à proposer des services d'apprentissage par exemple (Figure 42) ou encore le développement de cellules de calcul dédiées à l'apprentissage profond (GPUs spécialisées Deep Learning, TPUs).

Figure 42: Liste de différents services proposés par la plateforme Google Cloud. © Google Cloud

3.5.2 Le calcul distribué

La disponibilité de la puissance de calcul évolue de pair avec le besoin de traiter de plus en plus de données : le monde produit de plus en plus de données chaque jour. Afin de traiter toutes ces données, il existe deux méthodes : utiliser des nouveaux composants plus performants tels que le HPC (High Performance Computing) ou les GPUs ou partager le travail entre plusieurs machines. L'approche expliquée dans cette section est orientée sur les méthodes de partage de travail sur plusieurs machines et l'approche des nouveaux composants est conservée pour la partie sur les librairies informatiques où certaines d'entre elles permettent d'exécuter le code sur divers composants spécifiques.

On parle de calcul distribué lorsque le calcul est distribué sur plusieurs machines. Le calcul distribué, réparti ou partagé, est l'action de répartir un calcul ou un traitement sur plusieurs microprocesseurs et plus généralement toute unité centrale informatique. On parle de calcul distribué sur des clusters de calcul spécialisés qui ici sont alloués dans le cloud. Le calcul distribué nécessite des structures particulières de programmes comparativement au programme courant pour exécution sur une machine. Par exemple, la distribution de ces tâches doit être indépendante du nombre de machines puisque le nombre de machines doit pouvoir être augmenté suivant la charge de travail à fournir. Toutes les opérations historiques ne sont pas nativement possibles lors de calcul distribué et doivent être réécrites. Seul un sous-ensemble d'opérations est possible car elles doivent pouvoir s'exécuter en parallèle. Les opérations Map et Reduce sont les plus connues d'entre elles car elles ont été la base d'un des premiers frameworks historiques. L'opération Map (Figure 43) est une opération réalisée de manière indépendante et parallèle sur tous les éléments et l'opération Reduce (Figure 44) est une opération prenant en entrée deux éléments et retournant un seul élément qui appliquée successivement, retourne un élément pour toute la base de données. Les frameworks disponibles pour concevoir de tels programmes ont pour but de simplifier l'écriture de programme en proposant un ensemble de fonctions telles que les Map et Reduce qui s'exécutent facilement de manière distribuée.

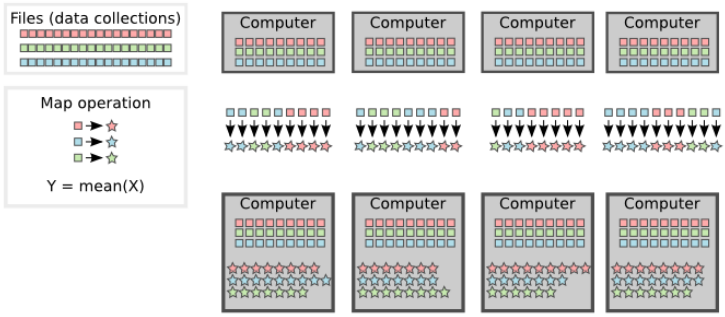

Figure 43: Illustration de l'application d'une fonction Map à un jeu de données.

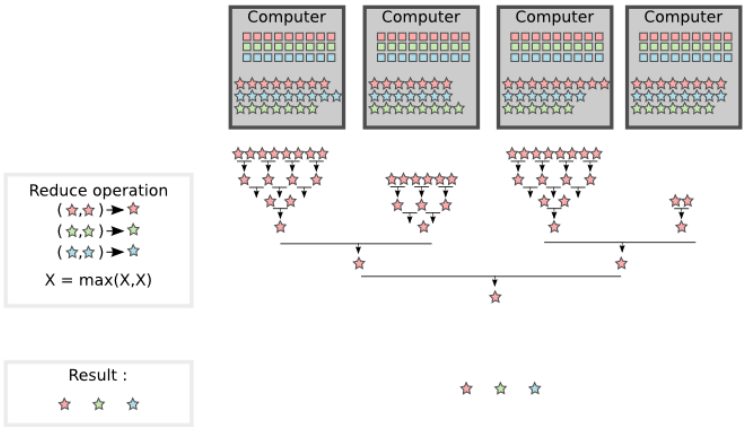

Figure 44: Illustration d'une fonction de Reduce sur un jeu de données.

L'utilisation classique de tels logiciels est de créer une version distribuée d'un programme qui s'exécute déjà sur une machine. La distribution d'un programme permet de pouvoir traiter des grandes quantités de données, capacité qui est désignée par le terme de scalabilité. En effet, l'un des avantages du calcul

distribué est l'élasticité horizontale, c'est-à-dire que la taille du cluster (nombre de machines) peut s'adapter à la charge de travail. La conception de tels algorithmes se fait couramment sous la forme de flot de données de manière à simplifier la distribution des calculs : en effet, ce patron de conception tire avantage du fait que les données sont stockées sur plusieurs machines. A la différence de décrire le programme comme une suite d'instructions agissant sur un ensemble de données de manière à calculer un résultat, un programme sous la forme de de flux de données va décrire le programme en utilisant des transformations sous forme de chaînes de traitement appliquées aux données : de cette façon, le programme connait toutes les chaînes de traitement à appliquer à ces données et peut donc les paralléliser les traitements indépendamment de la localisation des données.

Parmi les outils les plus connus, on peut trouver Apache Beam qui est un modèle de programmation unifiée open source pour définir et exécuter des traitements en flux de données. Beam est une surcouche qui permet d'exécuter le programme distribué sur plusieurs outils et le but principal est d'unifier les modes de conception de programme d'analyse de données distribuées. En particulier, Apache Beam est une implémentation du modèle de Dataflow qui est basée sur des travaux antérieurs sur des abstractions de traitements distribués de Google. Apache Kafka est un projet open source d'une plateforme de traitement streaming écrit en Java et Scala. Le projet avait également pour but de fournir un framework unifié et performant afin de permettre des analyses temps réel de flux (les données ne sont pas statiques et arrivent en continu). Pour conclure, le dernier framework à être cité ici est Apache Spark qui est sans doute le plus connu des environnements de conception de traitement distribué et sera décrit en détails dans une prochaine section.

3.5.3 Le stockage massif

Comme il a été évoqué précédemment, le point clé de tout système distribué est le stockage distribué de la donnée. La quantité de données à traiter a beaucoup augmenté : c'est pourquoi le traitement et le stockage ont dû être distribués (répartis sur plusieurs machines). En informatique, un système de fichiers distribués ou système de fichiers en réseau est un système de fichiers qui permet le partage de fichiers à plusieurs machines au travers du réseau informatique. L'accès aux fichiers est donc modifié et nécessite des outils et protocole adaptés. Le HDFS (Hadoop Distributed File System) est un système de fichiers distribué, extensible et portable développé par Hadoop à partir du GoogleFS. Écrit en Java, il a été conçu pour stocker de très gros volumes de données sur un grand nombre de machines équipées de disques durs banalisés. Il permet l'abstraction de l'architecture distribuée, afin de manipuler un système de fichiers distribué comme s'il s'agissait d'un disque dur unique. Une architecture de machines HDFS (aussi appelée cluster HDFS) repose sur deux types de composants majeurs :

- NameNode, qui gère les noms, l'arborescence du système de fichiers, les métadonnées des fichiers et des répertoires. Il centralise la localisation des blocs de données répartis dans le cluster.
- DataNode, stocke et restitue les blocs de données

Lors du processus de lecture d'un fichier, le NameNode est interrogé pour localiser l'ensemble des blocs de données. Le DataNode contenant le fichier indiqué par le NameNode prend le relai pour satisfaire la requête. Le HDFS permet de stocker des fichiers de grande taille sur plusieurs machines en les divisant en petits fichiers : il faut alors que le type de fichier le permette (un fichier texte peut diviser selon les lignes par exemple). De même, le système de fichiers distribué gère la fiabilité en répliquant les données sur

plusieurs hôtes. La valeur par défaut de réplication est 3 : les données sont stockées sur trois nœuds. La réplication permet au système d'être robuste aux défaillances des machines : c'est-à-dire que la donnée n'est pas perdue si une machine a une défaillance. Le système de fichiers distribués est l'élément essentiel de tout système distribué car la configuration en flot de données a pour but de déplacer les traitements au plus près de la donnée sur chaque machine. Mettre en place un cluster de machines pour faire de l'analyse de données requiert que les machines soient sur le même réseau pour pouvoir communiquer et qu'un système de fichier distribué soit mis en place sur le réseau de machines.

3.5.4 Apache Spark

Spark (ou Apache Spark) est un framework open source de calcul distribué. Développé à l'université de Californie à Berkeley par AMPLab, Spark est aujourd'hui un projet de la fondation Apache. Ce projet a pour but de fournir les outils essentiels pour des traitements relatifs au domaine du Big Data afin d'effectuer des analyses complexes à grande échelle. En 2009, Spark fut conçu par Matei Zaharia lors de son doctorat au sein de l'université de Californie à Berkeley. À l'origine, Spark est une solution pour accélérer le traitement des systèmes Hadoop et les développeurs ont rapidement pu démontrer la rapidité de leur produit en termes d'exécution des tâches par rapport à MapReduce. Aujourd'hui, le projet continue d'évoluer grâce à ses nombreux contributeurs qui participent à son développement et sont issus d'environ 200 sociétés différentes (exemples : Intel / Facebook / IBM / Netflix). Depuis 2015, plus de 1000 contributeurs ont été recensés.

Contrairement au système Hadoop qui écrit sur le disque le résultat de chaque opération, Spark réalise une lecture des données au niveau du cluster sur le système de fichier distribué, effectue toutes les opérations d'analyse nécessaires, puis écrit les résultats à ce même niveau. Le principe fondamental sur lequel est basé Spark est la RDD (Resilient Distributed Dataset) (Zaharia M., et al. 2012). Un RDD est une collection de données calculée à partir d'une source et conservée en mémoire vive (tant que la capacité le permet). Toutes les opérations sont connues grâce au graphe de calcul qui a été construit dans le programme et la demande d'une action de l'utilisateur (je veux calculer telle valeur, par exemple) déclenche l'exécution des opérations strictement nécessaires sur les différentes machines contenant les données requises. Ce graphe permet également d'optimiser l'exécution en parallèle et la communication entre machines. Une RDD est dite résiliente ou robuste à la défaillance (fault-tolerant) car le graphe de calcul permet également de recalculer en cas de perte de données au cours du traitement. L'ancêtre de Spark est le framework MapReduce qui a été désigné à la base pour exécuter les tâches par étape et écrire sur disque afin d'être robuste aux différentes défaillances. Spark exécute la totalité des opérations d'analyse de données en mémoire vive et en temps réel : il ne s'appuie sur des disques seulement lorsque sa mémoire vive n'est plus suffisante. Ce travail en mémoire permet de réduire les temps de latence entre les traitements, ce qui explique une telle rapidité. De ce fait, Spark peut travailler sur la totalité des données en même temps et être jusqu'à 100 fois plus rapide que les frameworks historiques (Figure 45). Cependant, Spark ne dispose pas de système de gestion de fichier qui lui est propre. Il est souvent utilisé avec HDFS qui reste actuellement la meilleure solution globale de stockage grâce à ses outils d'administration, de sécurité et de monitoring plus avancés.

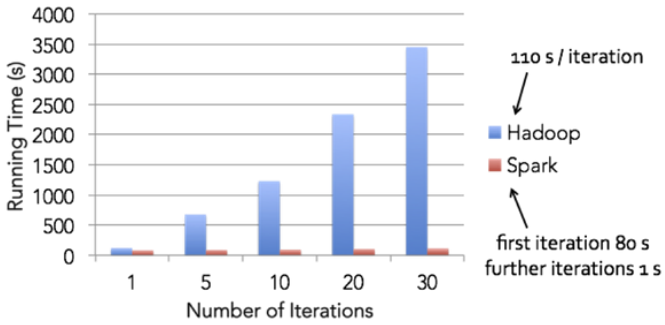

Figure 45: Exemple de comparaison de performance entre Apache Spark et le framework historique Hadoop. Source

3.5.5 Scikit-Learn

Scikit-Learn est une bibliothèque libre Python dédiée à l'apprentissage automatique. Elle est développée par de nombreux contributeurs notamment dans le monde académique par des instituts français d'enseignement supérieur et de recherche comme Inria et Télécom ParisTech.

Elle comprend notamment divers utilitaires (Figure 46) :

- Des classifieurs tels que les forêts aléatoires, des régressions logistiques, les machines à vecteurs de support, des algorithmes de Boosting…
- Des méthodes non supervisées telles que les KMeans, Mean Shift, …
- Des fonctions de pré-traitements de données telles que la normalisation, la discrétisation, un one hot encoder, et bien d'autres
- Elle est conçue pour s'intégrer avec les autres bibliothèques libres Python, notamment NumPy et SciPy.

L'idée principale de Scikit-Learn est de mettre à disposition des implémentations performantes des éléments classiques d'apprentissage automatique. La nouveauté qu'elle a apportée est de proposer une interface commune pour tous les classifieurs, de façon à pouvoir changer facilement y compris vers des classifieurs extérieurs à la librairie. De la même manière, cela a permis deux choses. D'une part, des outils d'optimisation d'hyperparamètres compacts et intuitifs par validation croisée (Random Search et Grid Search) sont apparus. Deuxièmement, cela a permis de construire un formalisme de chaîne de traitement qui consiste à enchainer des opérations : par exemple, plusieurs prétraitements de données sont appliqués avant d'apprendre le classifieur final. Peu à peu, l'interface est désormais le standard dans le domaine : elle a d'ailleurs été reprise par la librairie de Machine Learning de Spark et la plupart des grandes librairies de Machine et Deep Learning proposent souvent une interface compatible à celle de Scikit-Learn (Tensorflow, Xgboost). Depuis ces débuts, elle a été plébiscitée pour sa flexibilité, sa simplicité, et sa performance. L'objectif initial a donc été respecté puisqu'il était d'apporté de l'aide à la recherche en fournissant l'accès à des outils performants pour divers tâches.

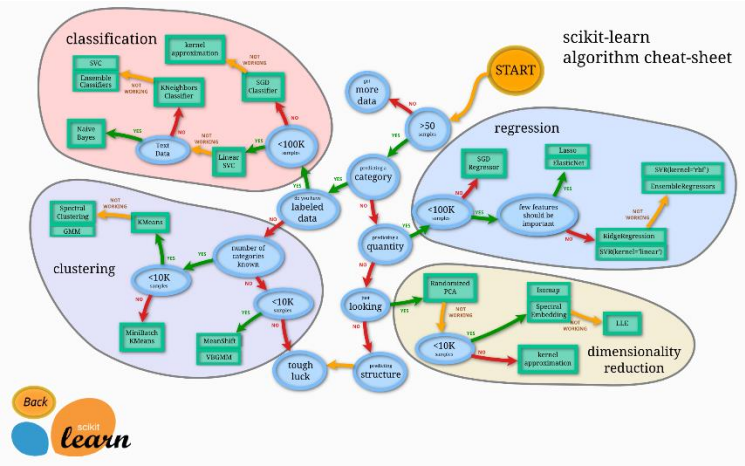

Figure 46: Bref aperçu des fonctions contenu dans scikit learn. © Scikit Learn

3.5.6 Tensorflow

TensorFlow (Abadi M., et al. 2016) est en simplifiant le pendant de Scikit-Learn pour l'apprentissage profond : cette librairie a été développée afin de simplifier la programmation et l'utilisation de réseaux profonds. La librairie est majoritairement maintenue par Google. Elle propose une interface simple et concise pour exprimer des algorithmes d'apprentissage et une implémentation efficace pour l'exécution de tels algorithmes. Un calcul exprimé à l'aide de TensorFlow peut être effectué avec peu ou pas de changement sur une grande variété de systèmes hétérogènes, allant des appareils mobiles tels que les téléphones et les tablettes aux systèmes distribués à grande échelle de plusieurs machines et des composants informatiques tels que les cartes GPU. Le système est flexible et est utilisé pour exprimer une grande variété d'algorithmes, y compris pour concevoir des modèles de réseaux de neurones profonds. Elle a été utilisée pour la recherche et le déploiement de l'apprentissage automatique dans une grande variété de domaines, contenant la reconnaissance vocale, la vision par ordinateur, la robotique, la récupération d'informations, le traitement du langage naturel, …

Le fonctionnement interne de Tensorflow est la clé de son succès. Un calcul TensorFlow est décrit par un graphe, qui se compose d'un ensemble de nœuds. Le graphe représente un flux de données et ses transformations, avec des extensions permettant à certains types de nœuds de maintenir et de mettre à jour des états persistants. Les différents clients (Python, C++, autres) permettent de générer ce type de graphe de calcul. Un exemple de code construisant un graphe de calcul puis l'exécutant est visible dans la Figure 47.

Les composants de TensorFlow sont :

- Le graphe de calcul, qui contient l'information de toutes les opérations qu'il est possible d'effectuer
- Chaque nœud correspond à une opération possédant zéro ou plusieurs entrées et de même pour la sortie.
- Une arête définit le sens suivant lequel circulent les tenseurs dans le graphe
- Un tenseur est un tableau de dimensionnalité non définie mais de type connu ou inféré depuis le graphe

- Une opération représente un calcul abstrait (par exemple, multiplication matricielle, ou addition). Elle possède un nom et des attributs éventuellement pour son exécution (exemple type des tenseurs)
- Un noyau est une implémentation particulière d'une opération exécutée sur un type particulier de périphérique (par exemple CPU ou GPU).
- Un binaire TensorFlow définit les opérations du graphe et leurs noyaux associés et disponibles.

Les programmes peuvent interagir avec le système TensorFlow en créant une Session durant laquelle un Run est appelé définissant les noms de variables à calculer ainsi qu'un ensemble de tenseurs optionnels pour alimenter les entrées éventuelles du graphe. En utilisant ces arguments, l'implémentation de TensorFlow peut calculer tous les nœuds qui doivent être exécutés afin de calculer les sorties demandées, et peut alors organiser leur exécution.

```
import tensorflow as tf

b = tf.Variable(tf.zeros([100]))                    # 100-d vector, init to zeroes
W = tf.Variable(tf.random_uniform([784,100],-1,1)) # 784x100 matrix w/rnd vals
x = tf.placeholder(name="x")                        # Placeholder for input
relu = tf.nn.relu(tf.matmul(W, x) + b)              # Relu(Wx+b)
C = [...]                                           # Cost computed as a function
                                                    # of Relu

s = tf.Session()
for step in xrange(0, 10):
  input = ...construct 100-D input array ...        # Create 100-d vector for input
  result = s.run(C, feed_dict={x: input})           # Fetch cost, feeding x=input
  print step, result
```

Figure 1: Example TensorFlow code fragment

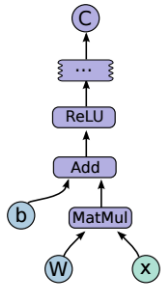

Figure 2: Corresponding computation graph for Figure 1

Figure 47: Exemple de génération de graphe de calcul à partir de code ©Tensorflow

La connaissance exacte et complète du graphe de calcul a de multiples impacts. Le premier est la facilitation de la parallélisation ou distribution de toutes ces opérations sur plusieurs supports du type CPU ou GPU ou entre machines. En effet, en déterminant la répartition entité/opération, il est facile de préparer les communications entre machines et les noyaux à utiliser suivant les supports. La deuxième conséquence est la possibilité de faire du calcul symbolique et donc de faire de la différenciation automatique. Si le gradient de chaque opération élémentaire est défini, il est possible de remonter automatiquement au gradient de chaque variable du graphe grâce à la règle de la chaîne. Et comme évoqué précédemment, cette différenciation automatique est un facteur primordial pour faire du Deep Learning et créer de nouvelles structures.

3.6 MÉTHODOLOGIE DE TRAVAIL

3.6.1 Applications.

Initialement, en termes d'applications, deux besoins ont été identifiés dans cette thèse : la détection de nuages et la classification des sols. De manière plus générale, le problème à résoudre est la classification appliquée aux images satellites. Le sujet principal est de développer une méthodologie qui permettra de résoudre les deux applications en ne changeant que les données en entrée. Les technologies d'apprentissage semblent permettre l'automatisation de telles tâches et la conception d'un tel système est logique avec les récents progrès. Pour être plus concret, le but n'est pas de résoudre le problème général de classification d'images mais la méthodologie doit être robuste à l'ajout d'une catégorie dans la classification des couverts ou au changement de satellites (en respectant un degré minimum de similarités).

3.6.2 Le Boosting distribué.

L'état de la recherche en début de thèse nous a poussé à croire que l'apprentissage automatique et des techniques telles que les Random Forest ou le Boosting pourraient résoudre les problèmes de classification d'images. Le premier axe a donc consisté à appliquer une méthode classique de Boosting au problème de détection de nuages. Plusieurs facteurs sont rentrés en jeu : premièrement, la conception des descripteurs numériques des images joue un rôle primordial dans les performances et deuxièmement, la taille de la base de données disponible a nécessité l'utilisation du calcul distribué pour l'analyser. La première contribution fut donc d'utiliser la puissance de calcul et la base de données disponible grâce à un algorithme de Boosting distribué. La quantité de données à notre disposition devrait permettre d'apprendre des classifieurs performants. De même, la recherche des descripteurs est un critère fondamental de performance et la puissance du cloud nous permet d'en considérer un nombre important dans notre analyse.

3.6.3 L'évolution rapide du Deep Learning intrigue.

La thèse a démarré en 2014 qui correspond aux premiers succès de l'apprentissage profond. En effet, les premiers réseaux profonds ont commencé à gagner des compétitions d'analyse de données à cette époque. A l'époque, peu d'outils existaient afin de les tester rapidement. Les outils et les méthodes se sont développés rapidement et Tensorflow et d'autres librairies ont vu le jour et permis la diffusion du Deep Learning au grand public. Les résultats affichés sur des problèmes classiques de traitement d'images ont fortement encouragé leur application aux images spatiales mais leur évolution rapide soulevait tout de même des interrogations. Il était alors naturel de tester ces méthodes à nos cas d'utilisation et le deuxième axe est donc issu de la réflexion et des expérimentations de telles techniques à nos cas d'utilisation.

3.6.4 Présentation du plan.

La partie suivante a pour but de proposer une nouvelle méthodologie d'apprentissage distribué. Dans un premier temps, une introduction à la méthodologie du Boosting est proposée. Une analyse des différentes stratégies de passage à l'échelle de cette méthodologie est ensuite réalisée. En particulier, nous examinons

comment la stratégie de distribution peut influencer les propriétés de passage à l'échelle. Enfin, l'analyse a permis d'identifier des points faibles dans les algorithmes existants et un nouvel algorithme de Boosting distribué est proposé dans la section 4.3. Des expériences sont menées pour assurer les propriétés désirées pour ce nouvel algorithme et les résultats et conclusion sont présentés dans la section 4.4.

La dernière partie s'attache à l'étude de l'application de l'apprentissage profond à l'image satellite et en particulier aux cas d'utilisation considérés dans cette thèse. Cette application nécessite une nouvelle formulation du problème pour appliquer des réseaux de neurones convolutionnels. Cette formulation est proposée dans 5.2. La section définit les éléments caractéristiques d'un apprentissage d'un réseau profond. En particulier, cette section décrit le patron de structure neuronale utilisée lors des apprentissages et décrit également les différentes étapes de traitements appliqués avant l'apprentissage. Les expériences et résultats des apprentissages sont présentés dans 5.4.

BOOSTING DISTRIBUÉ

Chapitre 4 BOOSTING DISTRIBUÉ

4.1 BOOSTING

4.1.1 Description générale de l'algorithme

Le principe du Boosting est basé sur une question posée dans les années 90 : un ensemble de classifieurs faibles peut-il créer un classifieur fort ? A cette époque, la puissance de calcul permettait facilement d'apprendre un classifieur faible mais était insuffisante pour apprendre un classifieur fort. Un classifieur faible est un classifieur dont la performance est légèrement supérieure à la performance du classifieur aléatoire[4]. Naturellement, la question de la création d'un classifieur fort à partir d'un ensemble de classifieurs est devenue importante. Dans un article de 1990, Robert Schapire a, pour la première fois, affirmé cette possibilité et proposé une méthode de construction de ce classifieur fort. Cette méthode a eu des répercutions significatives dans le monde du Machine Learning (Freund Y. et Schapire R. 1996) puisque la méthode est encore utilisée de nos jours.

Le principe de base du Boosting est d'attribuer des poids aux exemples de la base de données et de les utiliser lors de l'apprentissage (Freund Y. et Schapire R. 1996). Les poids associés aux exemples permettent de mesurer l'importance à leur accorder durant l'apprentissage. Les poids sont initialisés de manière uniforme de façon à considérer tous les exemples de la même manière au départ. L'apprentissage consiste à apprendre de manière itérative un ensemble de classifieurs faibles. Le classifieur fort est alors défini à partir d'un vote majoritaire sur cet ensemble. Un classifieur faible est appris à chaque itération puis rajouté à l'ensemble total. A chaque itération, la base de données est repondérée grâce aux prédictions du classifieur faible appris. En effet, les poids (« importance ») des exemples mal classifiés par le classifieur courant sont augmentés et inversement pour les exemples correctement classifiés. Ainsi, à chaque itération, l'apprentissage des classifieurs faibles se concentre sur les exemples les plus difficiles à classifier.

4.1.2 Classifieurs faibles

La méthode du Boosting est basée sur l'apprentissage de classifieurs faibles. Il convient donc de les définir. Les classifieurs faibles sont des classifieurs dont les performances sont faibles mais supérieures à l'aléatoire. Ils ont, en général, une formulation simple et sont peu coûteux à estimer sur une base de données d'exemples. Dans ce rapport, les classifieurs faibles sont notés h et la condition selon laquelle ils sont meilleurs que l'aléatoire pour une classification en N_c classes distinctes s'écrit :

$$\mathbb{E}[h(\boldsymbol{X}) = Y] > \frac{1}{N_c}$$

Qui s'écrit, pour le cas particulier, d'une classification binaire ($N_c = 2$) :

$$\mathbb{E}[h(\boldsymbol{X}) = Y] > 0.5.$$

La question soulevée est alors de déterminer les classifieurs qui peuvent être utilisés en pratique pour le Boosting. La littérature sur les applications du Boosting est conséquente mais certaines familles de

[4] Classifieur donnant des prédictions aléatoires

classifieurs faibles sont particulièrement utilisées. Les classifieurs bayésiens naïfs, les réseaux de neurones (du type perceptron multi couches) et les arbres de décision ont beaucoup été utilisés car leurs points communs sont leur simplicité d'apprentissage et leur faible performance en généralisation. Une des hypothèses sous-jacentes est que le problème à traiter est suffisamment compliqué pour que les classifieurs ne généralisent pas bien. En particulier, les arbres de décision à 1 niveau (autrement appelé « decision stump ») ont été beaucoup utilisés car ils sont faciles à apprendre et leur simplicité empêche le sur-apprentissage. Un arbre de décision à 1 niveau est une simple comparaison d'un attribut numérique avec un seuil. Supposons que j est l'indice de l'attribut numérique utilisé, $\gamma \in \mathbb{R}$ est le seuil, et un paramètre $p \in \{-1,1\}$ est rajouté afin de paramétrer le sens de la comparaison, alors le decision stump peut s'écrire :

$$h_{s,\gamma,p}(\boldsymbol{X}) = \begin{cases} 1 \; si \; p\boldsymbol{X}^j \geq p\gamma \\ 0 \; sinon \end{cases}$$

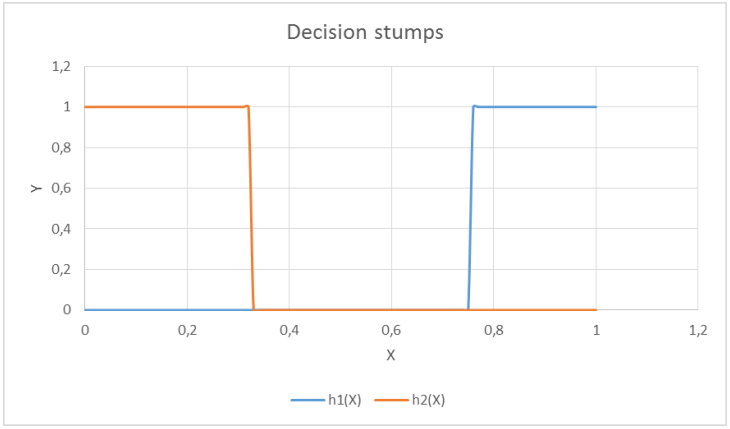

Figure 48: Graphe de la réponse de deux decision stumps de paramètres respectifs: (1, 0.75, 1) and (1, 0.33, -1)

L'apprentissage d'un « decision stump » est réalisé de la même manière que celui d'un arbre à un niveau et la méthode est décrit dans la partie des arbres de décision (cf 3.4.1.4). Pour chaque attribut numérique j, un tri croissant est d'abord réalisé sur les données pour obtenir $V^j = \left(v_i^j\right)_{i=1\ldots N}$. Toutes les valeurs possibles v_i sont testées et leur performance est évaluée sous la forme du gain d'information ΔI_i . Le meilleur seuil i^* est ainsi choisi et permet de définir le meilleur seuil γ^j et son gain d'information ΔI^j pour chaque variable j. Pour conclure, la meilleure feature j^* est choisie en fonction du gain d'information et permet enfin de renvoyer les paramètres associés γ, j^*. Le sens p est attribué à posteriori en étudiant la répartition des classes en fonction de la réponse du decision stump et nous verrons plus loin qu'il n'a pas d'importance.

Entrées :

- $S = \{(\boldsymbol{x}_i, y_i), i = 1, \dots, N\}$, base de données d'exemples en entrée

For $j \in \{1 \dots p\}$

// Tri par ordre croissant des valeurs numériques de l'attribut j

$V = (v_i)_{i=1\dots N} \leftarrow Tri\ par\ ordre\ coissant\ de\ \left(\boldsymbol{x}_i^j\right)_{i=1,\dots,N}$

// Calcul du gain d'information pour chaque seuil possible

For $i \in \{1 \dots N\}$

$S_l = (v_1, \dots, v_i)\ , S_r = (v_i + 1, \dots, v_m)$

$\Delta I_i = \Delta I(V, S_l, S_r) = I(V) - [I(S_l) + I(S_r)]$

// Sélection du meilleur seuil

$i^* = \underset{i}{\operatorname{argmax}}\, \Delta I_i$

$\gamma^j = \frac{v_{i^*} + v_{i^*+1}}{2}$

$\Delta I^j = \Delta I_{i^*}$

EndFor

// Sélection de la meilleur variable à utiliser

$j^* = \underset{j}{\operatorname{argmax}}\, \Delta I^j$

$\gamma = \gamma^{j^*}$

Renvoie $h_{j^*,\gamma,+1}$

Algorithme 1: Apprentissage d'un decision stump

4.1.3 Construction de l'ensemble

Le premier algorithme ou classifieur à utiliser la technique du Boosting est le classifieur Adaboost (pour Adaptative Boosting). Plusieurs améliorations ont été apportées à la méthodologie générale pour aboutir à la proposition de ce classifieur et de sa méthode d'apprentissage. Tout, d'abord, à des fins d'efficacité, Adaboost utilise une fonction de perte exponentielle. Pour une classification binaire dans $\{-1,1\}$, la fonction de perte s'écrit :

$$L(X, Y) = e^{-YX}.$$

La fonction de coût exponentielle est un majorant de la probabilité d'erreur de classification : sa minimisation permet donc de minimiser cette probabilité. De plus, l'optimisation de cette fonction de coût induit également la maximisation de la marge (Schapire R., et al. 1998) de la même manière que les SVMs.

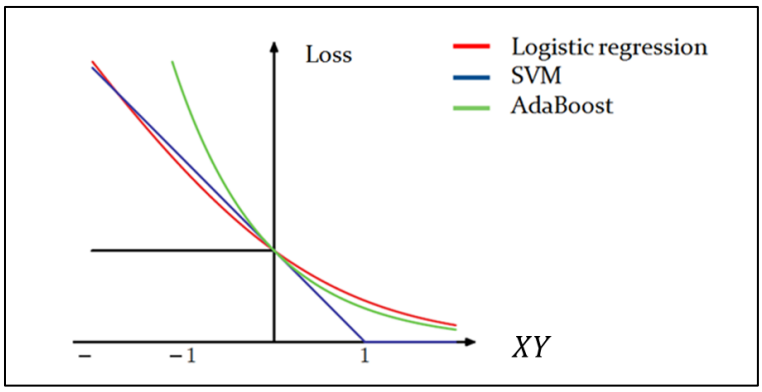

Figure 49: Graphique représentant les réponses des différentes fonctions de perte aux mêmes entrées. L'échelon noir est l'erreur de classification. Source

Supposons que $S = \{(\boldsymbol{x}_i, y_i), i = 1, \dots, N\}$ est la base d'entrainement. Un algorithme de Boosting nécessite la définition de poids associés à chaque exemple : $D_t = \{w_i^t, i = 1, \dots, N\}$. Ces poids sont utilisés pour calculer le risque empirique pondéré :

$$\epsilon(S, f, D_t) = \sum_{i=1}^{N} w_i^t \, \mathbb{I}(f(\boldsymbol{x}_i) \neq y_i).$$

Le principe du Boosting est de construire une fonction de décision comme une combinaison linéaire de classifieurs faibles h_t. T désigne le nombre de classifieurs faibles utilisés. Ici, nous utiliserons les decision stumps présentés précédemment comme classifieurs faibles, c'est-à-dire,

$$f(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})$$

A chaque itération, l'algorithme apprend un nouveau classifieur faible h_t dont le risque empirique pondéré est $\epsilon\,(S, h_t, D_t)$ et l'ajoute dans sa fonction de décision finale avec un coefficient α_t qui reflète les performances du classifieur faible. En suivant, les poids w_i^t sont mis à jour selon la difficulté pour classifier l'exemple $\boldsymbol{x}_i$. En d'autres termes, les poids utilisés pour la prochaine itération seront augmentés pour les exemples classifiés incorrectement et inversement, diminués pour les exemples correctement classifiés.

Entrées :

- $S = \{(\boldsymbol{x}_i, y_i), i = 1, \dots, N\}$, base de données d'exemples en entrée
- T Nombre d'itérations d'Adaboost

$t \leftarrow 0$

$\forall\, i = 1, \dots, N, w_i^0 = \frac{1}{N}$

While t < T

$h_t \leftarrow$ Decision stump appris sur S et D_t en utilisant Algorithme 1

// Calcul de l'importance du classifieur faible

$$\alpha_t = 0.5\ \log\frac{1 - \epsilon\ (S, h_t, D_t)}{\epsilon\ (S, h_t, D_t)}$$

// Mise à jour des poids

$$\forall\, i = 1, \dots, N, \mathrm{w_i^{t+1}} = \frac{\mathrm{w_i^t}}{Z_{t+1}} \exp[-\alpha_t y_i\ h_t(\boldsymbol{x}_i)]$$

// Définition du facteur de normalisation Z_{t+1}

$$Z_{t+1} = \sum_{i=1}^{N} w_i^t\ \exp[-\alpha_t y_i\ h_t(\boldsymbol{x}_i)]$$

endWhile

Algorithme 2: Algorithme d'Adaboost

Après avoir présenté l'algorithme Adaboost, il convient de faire les remarques suivantes :

1. $\forall\, i = 1, \dots, N, w_i^0 = \frac{1}{N}$, les poids sont d'abord initialisés de manière uniforme sur le jeu de données.
2. $\alpha_t = 0.5\ \log\frac{1-\epsilon\ (S,h_t,D_t)}{\epsilon\ (S,h_t,D_t)}$ Le coefficient d'importance de h_t dans le classifieur final est calculé à partir de la performance de généralisation du classifieur faible appris. A noter que α_t peut corriger le sens de classification : si $\epsilon\ (S, h_t, D_t) = 1 - \epsilon\ '(S, h_t, D_t)$ alors $\alpha_t = -\alpha_t'$. C'est pourquoi on définit conventionnellement souvent la frontière (edge) d'un classifieur faible : $\epsilon\ (S, h_t, D_t) = 0.5 - \gamma(S, h_t, D_t)$ et $\alpha_t = 0.5\ \log\frac{0.5+\gamma\ (S,h_t,D_t)}{0.5-\gamma(S,h_t,D_t)}$. L'évolution de α_t en fonction du risque empirique pondéré est visible sur la Figure 50.
3. $\forall\, i = 1, \dots, N, \mathrm{w_i^{t+1}} \propto \mathrm{w_i^t} \exp(-\alpha_t y_i\ h_t(x_i))$ donc les poids sont augmentés d'un facteur multiplicatif e^{α_t} lorsque le classifieur se trompe i.e. pour $y_i\ h_t(x_i) = -1$ et inversement.
4. Z_{t+1} est calculée afin de normaliser la distribution discrète

Enfin, l'ajout de classifieurs faibles supplémentaires dans le classifieur final est assuré de contribuer à la baisse de l'erreur d'entrainement (Freund Y. et Schapire R. 1996). Cependant, afin d'éviter le sur-apprentissage, la taille du classifieur final est contrôlée et T est choisi par validation croisée.

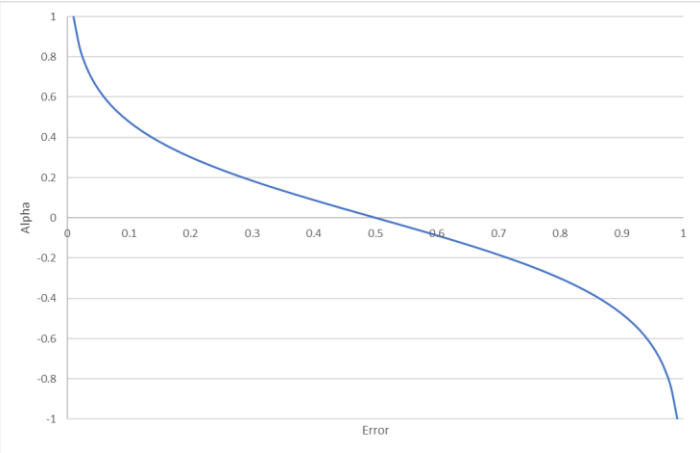

Figure 50: Valeur du coefficient alpha pour différentes valeurs d'erreur de généralisation d'un classifieur.

4.1.4 Echantillonnage préférentiel

La pondération de la base de données c'est-à-dire le choix des poids à chaque itération est un facteur essentiel pour la convergence d'Adaboost mais cette pondération peut être utilisée de plusieurs manières. Deux stratégies principales sont utilisées : la repondération et le rééchantillonnage. La repondération est la technique la plus classique pour traiter une base pondérée : les poids sont directement utilisés dans la fonction de coût. Cependant, certains algorithmes ne sont pas adaptés pour traiter des poids associés aux exemples et doivent donc être modifiés. Par exemple, lors d'une construction d'un arbre de décision, la condition d'arrêt est que les feuilles contiennent au maximum un certain nombre d'exemples pour éviter le surapprentissage et maintenir la cohérence statistique de la feuille. Cette condition doit être modifiée pour que les feuilles contiennent au minimum un pourcentage de la distribution, i.e., du poids total mais cette condition n'a plus de signification statistique. La principale critique de cette approche est qu'elle n'est pas très efficace en termes de calcul : tous les exemples sont considérés alors qu'une grande partie ont des poids très proches de zéro. A l'opposé, après quelques itérations, tout le poids de la base de données se concentre sur quelques exemples (les plus durs à classifier) : ce phénomène est connu et explique le surapprentissage souvent observé lors de l'utilisation d'Adaboost. Une des solutions à ce problème est l'utilisation d'une technique de rééchantillonnage ou échantillonnage préférentiel (Bradley J. et Schapire R. 2007) (Dubout C. et Fleuret F. 2014) (Kalal Z., Matas J. et Mikolajczyk K. 2008). Les poids attribués aux exemples sont utilisés pour définir une distribution de loi de probabilité sur la base de données et la génération d'un ensemble d'exemples selon cette loi est utilisée à chaque itération pour apprendre le nouveau classifieur. De nouveaux poids sont associés aux exemples extraits pour prendre en compte de l'échantillonnage et garantissent la représentativité de ces exemples par rapport à la base de données totale. Cet échantillonnage permet de pallier les phénomènes cités précédemment à savoir le fait d'avoir des poids proches de zéro et le sur-apprentissage.

Extraction d'un échantillon $\tilde{S}$ de S et $\mathrm{D_t}$ à partir d'une loi d'échantillonnage g_θ (qui sera précisé ulterieurement)

Calcul de $\widetilde{\mathrm{D_t}}$, la nouvelle distribution sur la base extraite

Apprentissage du décision stump h_t sur $\tilde{S}, \widetilde{D_t}$ avec Algorithme 1

Algorithme 3: Echantillonnage préférentiel pour l'apprentissage d'un decision stump

L'explication de la méthode de sous échantillonnage préférentiel nécessite de rappeler que le but de l'apprentissage d'un classifieur faible dans Adaboost est de minimiser le risque empirique suivant :

$$\epsilon(h, \mathrm{S}, \mathrm{D_t}) = \sum_{i=1}^{N} w_i^t \, \mathbb{I}(h(\boldsymbol{x}_i) \neq y_i).$$

Cette quantité peut être approchée grâce à un sous-échantillon de la base de données. Supposons que l'on extrait un ensemble $\tilde{S} = \left(\boldsymbol{x}_{u_j}, y_{u_j}\right), j = 1, \dots, M$ où $\left((u_j), j = 1 \dots M\right)$ désigne les indices des éléments présents dans le sous-échantillon. Supposons également que le tirage se fasse de la manière suivante :

$$\mathbb{P}\left[(\boldsymbol{x}_i, y_i) \in \tilde{S}\right] = g_\theta\big((\boldsymbol{x}_i, y_i), w_i\big).$$

En d'autres termes, la probabilité qu'un élément $(\boldsymbol{x}_i, y_i)$ de la base de données initiale soit inclus dans $\tilde{S}$ est égale à $g_\theta\big((\boldsymbol{x}_i, y_i), w_i\big)$. On définit $\hat{\epsilon} = \epsilon\big(h, \tilde{S}, \widetilde{D_t}\big)$ où $\widetilde{D_t} = \big(\tilde{w}_{\mathrm{j}}^{\mathrm{t}}\big), j = 1, \dots, M$ et :

$$\forall j = 1, \dots, M, \tilde{w}_{\mathrm{j}}^{\mathrm{t}} = \frac{w_{u_j}^t}{g_\theta\left(\left(x_{u_j}, y_{u_j}\right), w_{u_j}^t\right)}.$$

On a alors : $\mathbb{E}_{\tilde{S}}[\,\hat{\epsilon}] = \epsilon(h, S, D_t)$. L'estimation n'est donc pas biaisée et le classifieur est appris en minimisant le risque empirique estimé.

$$h_t = \underset{h \in \mathcal{H}}{\operatorname{argmin}}\ \hat{\epsilon}.$$

La loi g_θ est appelée loi de proposition puisqu'elle « propose » des exemples avec une certaine probabilité. Différentes lois de proposition d'exemples pour le sous échantillonnage sont utilisées dans les applications du Boosting (Bradley J. et Schapire R. 2007) (Dubout C. et Fleuret F. 2014) (Kalal Z., Matas J. et Mikolajczyk K. 2008) :

- Bootstrap :

$$g_\theta\big((X, Y), \mathrm{w}\big) = \frac{1}{N}$$

C'est une loi de proposition uniforme qui propose des exemples sans tenir compte de leur importance.

- Weighted sampling :

$$g_\theta\big((X,Y),w\big) = w$$

C'est une loi de proposition directement liée à l'importance des exemples (Dubout C. et Fleuret F. 2014) (Kalal Z., Matas J. et Mikolajczyk K. 2008).

L'utilisation de l'échantillonnage préférentiel diffère de la formulation classique d'Adaboost et permet d'améliorer les performances et la stabilité de l'algorithme.

4.1.5 Forces et faiblesses

Le Boosting, et en particulier l'algorithme d'Adaboost, sont souvent utilisés car ils offrent beaucoup d'avantages. Le premier avantage est qu'il est simple à implanter ce qui est important lorsque le but est de créer une version distribuée de cet algorithme. De plus, comme il a déjà été évoqué, la fonction de perte exponentielle permet de maximiser la marge comme les SVMs (Schapire R., et al. 1998) avec un coût d'optimisation plus faible. La méthode Adaboost est également connue pour sa faculté à sélectionner les variables efficacement : Adaboost est souvent utilisé en présence d'un nombre important de variables. Un classifieur faible utilise en général peu de variables et l'analyse des variables utilisées par les classifieurs faibles de l'ensemble boosté permet de sélectionner les features efficacement (Viola P. et Jones M. 2001). L'utilisation des « decision stumps » augmente cette faculté de sélection et évite le sur-apprentissage (Viola P. et Jones M. 2001): leur structure ne permet d'encoder des phénomènes complexes qui se généralisent souvent mal.

Cependant, un phénomène de sur-apprentissage est souvent observé lors de l'utilisation d'Adaboost. En effet, le classifieur a une erreur très faible sur sa base d'entrainement mais ces performances ne se généralisent pas bien à des exemples qu'il n'a pas vus. Comme il a déjà été remarqué auparavant, la méthode se focalise sur les exemples difficiles à classifier de sa base de données d'entrainement ce qui explique le sur-apprentissage et la sensibilité au bruit de classification. La plupart de ces faiblesses peuvent être corrigées en augmentant la quantité de données à traiter et une implantation distribuée de l'algorithme prend alors toute son importance pour ne pas avoir des temps de calcul rédhibitoires.

4.1.6 Variantes

Il existe une multitude de variantes d'Adaboost qui essaient de palier ses faiblesses en modifiant la formulation de l'algorithme.

- La première modification majeure apporté à Adaboost fut la mise à jour des poids (Schapire et Singer 1999) (Friedman J., Hastie T. et Tibishirani R. 2001) afin de prendre en compte la confiance du classifieur en plus de la prédiction. Supposons que le classifieur faible appris h_t possède une fonction de confiance g_t c'est-à-dire un score de classification tel que :

  $$h_t(\boldsymbol{x}) \in \{-1,1\}, f_t(\boldsymbol{x}) \in [-1,1], h_t(\boldsymbol{x}) = 1 \; si \; f_t(\boldsymbol{x}) \geq 0 \; et \; 0 \; sinon$$

  Alors elle peut être utilisée pour mettre à jour les poids de manière à incorporer le degré de confiance que le classifieur a attribué à l'exemple. Le poids d'un exemple classifié incorrectement avec un grand degré de confiance doit être augmenté de manière plus importante que celui d'un exemple classifié incorrectement avec un degré de confiance inférieure. La mise à jour des poids s'écrit alors :

$$\forall\, i = 1, \dots, N, \mathrm{w}_\mathrm{i}^{\mathrm{t+1}} = \frac{\mathrm{w}_\mathrm{i}^{t}}{Z_{t+1}} \exp[-\alpha_t y_i\, f_t(\boldsymbol{x}_i)]$$

- La deuxième évolution majeure fut l'ajout d'un terme de régularisation dans la fonction de perte. Comme il est traditionnellement réalisé pour des régressions linéaire et d'autres classifieurs, un terme de régularisation sous la forme d'une norme 2 des coefficients d'importance des classifieurs est ajoutée à la fonction de perte. D'autres types de régularisation (en particulier, d'autres normes) peuvent également utilisées.

$$f(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}), \boldsymbol{\alpha} = (\alpha_t)_{t=1\dots T}$$

$$\epsilon(S, f) = \frac{1}{N}\sum_{i=1}^{N} L(f(x_i), y_i) + \lambda\, \Omega(f) = \frac{1}{N}\sum_{i=1}^{N} e^{-y_i f(x_i)} + \lambda\, \|\boldsymbol{\alpha}\|_2.$$

- La dernière évolution majeure du Boosting fut de généraliser la procédure à n'importe quelle fonction de perte (et pas uniquement la fonction exponentielle). Le principe de Boosting et de l'utilisation de classifieurs faibles a été généralisé à n'importe quelle fonction de perte dérivable (Gradient Boosting (Natekin A. et Knoll A. 2013), Anyboost). En effet, un développement limité à l'ordre 2 permet de simplifier le problème d'optimisation qui peut alors être résolu par l'apprentissage d'arbre de décision de manière itérative (Boosting). Pour rappel, la méthodologie du Boosting consiste à apprendre de manière itérative des classifieurs qui sont ajoutés au classifieur global. Si l'on note f_t le classifieur final à la fin de l'itération t, alors f_t est obtenu en ajoutant le classifieur faible pondéré $k_t = \alpha_t\, h_t(\boldsymbol{x})$ appris durant l'itération t au classifieur f_{t-1} :

$$f_t(\boldsymbol{x}) = \sum_{l=1}^{t} \alpha_l h_l(\boldsymbol{x}) = f_{t-1}(\boldsymbol{x}) + k_t(\boldsymbol{x}),$$

Durant l'itération t, le classifieur k_t est obtenu en minimisant le risque empirique $\epsilon(S, f_t)$ qui peut être simplifié en utilisant un développement à l'ordre 2 de la fonction de coût.

$$\epsilon(S, f_t) = \frac{1}{N}\sum_{i=1}^{N} L(f_t(\boldsymbol{x}_i), y_i) \approx \frac{1}{N}\sum_{i=1}^{N} L(f_{t-1}(\boldsymbol{x}_i), y_i) + g_i\, k_t(\boldsymbol{x}_i) + 0.5 h_i k_t(\boldsymbol{x}_i)^2$$

Où $g_i = \frac{\partial L}{\partial X}\Big]_{(f_{t-1}(x_i), y_i)}$ et $h_i = \frac{\partial^2 L}{\partial X^2}\Big]_{(f_{t-1}(x_i), y_i)}$. Alors le problème de minimisation devient le suivant en supprimant les constantes de la formule précédente :

$$\frac{1}{N}\sum_{i=1}^{N} g_i\, k_t(\boldsymbol{x}_i) + 0.5 h_i k_t(\boldsymbol{x}_i)^2$$

Le classifieur k_t est choisi en minimisant la quantité précédente qui ne dépend seulement des formules de la dérivée et la dérivée seconde de la fonction de coût. N'importe quelle fonction de coût dont les formules de la dérivée et la dérivée seconde sont connues peut être utilisée avec la méthode du Gradient Boosting. Cette méthode est actuellement beaucoup utilisée et a notamment remporté beaucoup de compétitions dans le domaine de l'analyse de données : elle fait partie des rares méthodes dont la fonction de perte est paramétrable. L'algorithme de Gradient Boosting ne tire donc aucun avantage de la forme de la fonction de perte pour simplifier

les calculs. En particulier, des efforts se sont également concentrés pour reformuler un algorithme de Boosting pour des fonctions de coût classiques comme la fonction logistique ou la fonction de Hinge. En particulier, l'algorithme a été modifié de manière à utiliser la fonction logistique (Bradley J. et Schapire R. 2007) (Collins M., Schapire R. et Singer Y. 2002).

De nombreuses modifications de l'algorithme de base Adaboost qu'il est difficile d'énumérer à cause de leur nombre ont été proposées mais le schéma de l'algorithme reste similaire à celui d'Adaboost. Une implantation distribuée d'Adaboost pourra intégrer les modifications nécessaires a posteriori sans changer l'architecture générale.

4.2 PASSAGE A L'ECHELLE : ETAT DE L'ART SUR LA DISTRIBUTION D'ADABOOST

La problématique est d'adapter la formulation actuelle d'Adaboost pour une exécution distribuée de manière à passer à l'échelle. On désigne par passer à l'échelle l'évolution des performances du classifieur en fonction de l'augmentation de la taille de la base de données ou encore du nombre de machines dans le cluster de calcul. L'impact mémoire des algorithmes n'est pas pris en compte dans notre cas d'étude au contraire de certains travaux (Lassale 2015). En effet, dans notre cas, l'utilisation du cloud nous permet d'adapter la puissance de calcul et la mémoire au besoin par l'allocation d'un nombre adéquat de machines et ainsi nous soustraire de cette contrainte. Au contraire, les travaux de (Lassale 2015) proposent des algorithmes qui s'adaptent à la ressource disponible.

La partie suivante décrit donc les différentes méthodes proposées pour distribuer les calculs d'Adaboost sur plusieurs machines. Les méthodes ont été classées suivant le type de données qui sont distribuées sur le cluster : des méthodes sont basées sur la distribution des paramètres du modèle sur le cluster, d'autres sur la distribution des données sur le cluster ou de la combinaison des deux.

Nom de la méthode	Quantité distribuée	
	Données	Paramètres
Best CV		X
Planet/Xgboost	X	
Distboost	X	
Adasampling	X	
Sybil	X	X
Multiboosting	X	X
Méthode proposée	X	

Tableau 3: Répartition par schéma de distribution

4.2.1 Les méthodes « parameter parallel »

Les méthodes « parameter parallel » parallélisent le calcul des différents paramètres du modèle sur la même base de données. En d'autres termes, un nœud de calcul a accès à la totalité des données mais pas à la totalité du modèle.

- Les premières méthodes de ce type sont les méthodes basées sur la recherche des hyper paramètres. Les hyper paramètres sont généralement des paramètres de structure du modèle qui ne peuvent être estimés par apprentissage. Ils sont habituellement sélectionnés par validation croisée. La recherche d'hyperparamètres est donc coûteuse en calculs car elle requiert beaucoup d'apprentissages différents pour la validation croisée. Cependant, les apprentissages peuvent facilement être parallélisés puisqu'ils sont indépendants.
- Une seconde stratégie courante consiste à paralléliser les calculs par variables ou famille de variables. La performance de ces méthodes de parallélisation est souvent améliorée en utilisant un grand nombre de variables. Parfois, seulement une partie de ces variables est effectivement utilisée mais les calculs sont réalisés pour toutes les features durant l'apprentissage. Certains apprentissages comme celui d'un arbre de décision permettent la parallélisation par variable : la meilleure décision est calculée par variable, puis sélectionnée entre variables par la suite. Le calcul de la meilleure décision par variables peut alors être réalisé en parallèle. Par exemple, pour la détection de visages, des milliers de features sont considérées (Viola P. et Jones M. 2001): les nœuds de calcul sont donc chacun responsable d'un sous ensemble de variables regroupées par similarité (Abualkibash M., El Sayed A. et Mahmood A. 2013).

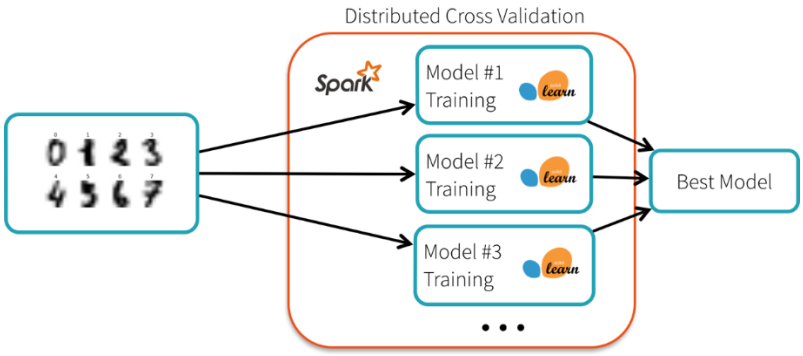

Figure 51: Exemple d'une structure de programme basé sur Spark et Scikit learn pour faire de la validation croisée distribuée. Source

4.2.2 Les méthodes « data parallel »

Les méthodes « data parallel » utilisent la distribution physique des données sur le réseau pour leurs calculs. En effet, le volume de données est souvent trop important pour être contenu sur une machine et il est donc partitionné et stocké sur plusieurs machines. L'idée est de réaliser les calculs localement sur des machines qui possèdent une partie des données. Des statistiques sont calculées sur les données et rassemblées pour calculer des paramètres. Le point important qui différencie les méthodes « data-parallel » et les méthodes « hybrides » est le fait que à tout instant, chaque sous partie des données (partition) est traitée en utilisant le modèle entier et non une partie du modèle comme les méthodes hybrides.

4.2.2.1 Adasampling

Adasampling (Cooper J. et Reyzin L. 2014) a été développée dans le but de diminuer au maximum les communications entre machines. Le Boosting a l'avantage de mesurer la difficulté de classifier les exemples au travers des poids associés à chaque exemple (voir l'Algorithme 2). Le principe d'Adasampling est d'extraire de chaque partition de données les exemples difficiles à classifier pour apprendre un modèle localement sur une machine. Pour cela, un modèle de Boosting est appris localement sur chaque machine pour chaque partition et permet d'extraire les exemples difficiles par partition qui sont rassemblés sur une machine centrale. Un modèle final de Boosting est appris sur la machine centrale et la base de données d'exemples difficiles et constitue le classifieur final. L'hypothèse de la méthode Adasampling est que l'information est entièrement contenue dans les exemples difficiles.

4.2.2.2 R/D-sampling

L'extraction d'exemples de partition de données est également beaucoup utilisée comme stratégie de distribution des calculs. En effet, l'utilisation de l'échantillonnage préférentiel a beaucoup d'avantages et en particulier, celui de permettre un apprentissage robuste sur une plus petite base de données. Le principe est alors d'extraire un certain nombre d'exemples de chaque partition de données en utilisant une stratégie d'échantillonnage pour former une plus petite base de données d'exemples représentative qui sera traitée localement sur une machine centrale. En particulier, le nombre d'exemples extraits par partition est calculé pour que la base de données extraite puisse être traitée localement sur une machine centrale. Cette base de données représentative est utilisée afin d'apprendre le classifieur faible de la procédure de Boosting sur une machine centrale. Les stratégies d'échantillonnage utilisées pour l'extraction d'exemples des partitions varient suivant les approches et les besoins (Bradley J. et Schapire R. 2007) (Dubout C. et Fleuret F. 2014) (Kalal Z., Matas J. et Mikolajczyk K. 2008) (Viola P. et Jones M. 2001) mais majoritairement, les exemples sont échantillonnés suivant leur poids dans la procédure de Boosting. Cependant, la plupart de ces stratégies ont été modifiées de manière à extraire les exemples pertinents par partition sans avoir accès à la base de données globales.

4.2.2.3 Distboost

D'autres approches n'utilisant pas d'extractions d'exemples ont été beaucoup utilisées. En particulier, la méthode Distboost (Lazarevic A. et Obradovic Z. 2001) n'extrait pas d'exemples des partitions mais apprend un classifieur par partition. Distboost suppose que le classifieur faible utilisé dans la procédure de Boosting est un ensemble constitué de plusieurs classifieurs appris sur les différentes partitions de données. Ainsi, à chaque itération du Boosting, un classifieur est appris sur chaque partition en utilisant les poids des exemples générés par le Boosting. Ces classifieurs sont transmis à la machine centrale qui construit le classifieur faible du Boosting en utilisant la moyenne de ces classifieurs et transmet le classifieur faible ainsi construit aux autres machines pour continuer la procédure de Boosting.

4.2.2.4 Xgboost

Les méthodes Xgboost et Planet (Google Planet (Panda B., et al. 2009), Xgboost (Chen T. et Guestrin C. 2016)) sont basées sur une implantation distribuée de l'apprentissage d'un arbre de décision. En effet, la

plupart des algorithmes de Boosting sont utilisés avec des arbres de décision comme classifieurs faibles et pourraient bénéficier d'une exécution distribuée de l'apprentissage d'un arbre de décision. Planet est un algorithme distribué d'apprentissage d'arbre de décision principalement basé sur la structure Map reduce pour l'exécution distribuée. Apparu en 2016, Xgboost (Chen T. et Guestrin C. 2016) est une version plus avancée de l'algorithme initialement proposé par Planet (Panda B., et al. 2009) qui introduit différentes optimisations nécessaires pour l'exécution distribuée non couvertes par l'algorithme initial. Xgboost est actuellement l'implantation distribuée du Boosting la plus utilisée.

Le principe de la méthode est le suivant. Lors de l'apprentissage local d'un arbre de décision, chaque étape consiste à trouver la meilleure séparation pour chaque feuille d'un arbre : comme il a déjà été présenté, cette recherche est basée sur un tri des valeurs par attribut qui ne convient pas à une exécution distribuée. Ce tri par valeur est alors remplacé par un calcul d'histogramme à l'intérieur de chaque feuille de l'arbre. Ces histogrammes sont alors utilisés pour choisir la meilleure séparation pour chaque feuille de l'arbre. Des histogrammes partiels sont calculés pour chaque feuille sur chaque partition et sommés par feuille pour obtenir l'histogramme total sur la base de données pour chaque feuille. Pour pouvoir sommer les histogrammes, il faut que les seuils utilisés soient fixés à l'avance. En pratique, les différents seuils pour les histogrammes sont choisis grâce à des histogrammes équirépartis (Panda B., et al. 2009) (Chen T. et Guestrin C. 2016) ou par un apprentissage comme le propose (Cooper J. et Reyzin L. 2014). Les seuils utilisés par des modèles de Boosting appris localement sur les partitions de données sont des bons candidats pour des arbres de décision appris sur la globalité des données (Cooper J. et Reyzin L. 2014). Les seuils sont soit calculés une fois au début de l'apprentissage ou pour chaque étape de croissance de l'arbre.

4.2.3 Les méthodes hybrides

Les méthodes hybrides parallélisent les calculs pour chaque partition de données et pour différents sous ensemble de paramètres. La quantité de données empêche l'exécution locale des algorithmes mais la quantité de paramètres empêche également que l'apprentissage soit fait pour tous les paramètres en même temps. De ce fait, les calculs sont séparés par partition de données et sous ensemble de paramètres. Le point important qui différencie les méthodes « data-parallel » et les méthodes « hybrides » est le fait que à tout instant, chaque sous partie des données est traitée en utilisant une sous partie du modèle et non le modèle entier comme les méthodes hybrides.

4.2.3.1 Multiboosting

La première approche, nommé Multiboosting (Webb 2000), consiste à apprendre en parallèle des modèles de Boosting sur chaque partition de données et ces modèles sont rassemblés dans un ensemble final dont la prédiction est un vote majoritaire. L'ensemble final est un ensemble dit de Bagging, qui a pour vocation à corriger les erreurs de sur-apprentissage de ses classifieurs internes appris sur les partitions en moyennant leur prédiction. Cette approche réduit considérablement les besoins de communications entre machines du cluster puisque les modèles appris sur chaque partition sont appris en parallèle sans communications entre machines. Cette méthode est une méthode hybride puisque les modèles appris sur chaque partition n'utilisent pas les autres modèles ni les autres partitions.

4.2.3.2 Parallel Boosting

Parallel Boosting est une méthode hybride utilisée par Google (Sybil[5]) (Collins M., Schapire R. et Singer Y. 2002) . Cette technique se base sur une variante parallèle du Boosting où tous les classifieurs et leurs coefficients sont mis à jour en même temps à chaque itération. Un ensemble initial de classifieurs fixes est cependant nécessaire. Initialement, tous les classifieurs sont ajoutés au comité final avec un poids arbitraire.

A chaque itération, pour chaque classifieur, la somme des poids où la prédiction du classifieur est correcte est calculée de même que la somme des poids où la prédiction du classifieur est fausse. Ces sommes sont ensuite utilisées pour mettre à jour le coefficient d'importance associé au classifieur simultanément pour chaque classifieur (Collins M., Schapire R. et Singer Y. 2002). Le calcul des sommes est naturellement parallélisable puisque des sommes partielles sont calculées par partition puis sommées pour obtenir les sommes globales. La méthode est hybride car le calcul est exécuté en parallèle pour chaque classifieur.

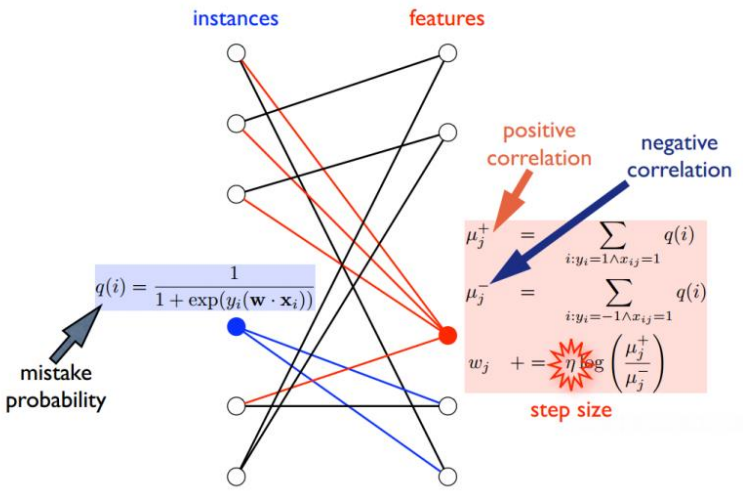

Figure 52: Schéma expliquant le principe du parallel boosting. Source

4.2.3.3 Parameter Server

La dernière méthode hybride de cette partie est une méthode plus générale que le Boosting qui mérite d'être mentionnée car elle est utilisée dans un nombre important d'algorithmes distribués. La méthode de Parameter Server (Li M., et al. 2013) est une implantation distribuée d'une descente de gradient asynchrone sur plusieurs partitions de données. Elle est utilisée pour la distribution de diverses techniques d'apprentissage comme la descente de gradient pour les réseaux de neurones. Son principe se rapproche des principes qui sont mis en place dans les méthodes distribuées hybrides (en particulier Sybil). Le principe est le suivant : chaque nœud de calcul calcule le gradient du modèle sur ses données de manière asynchrone et renvoie cette valeur au nœud central, le Parameter Server. Le Parameter Server réalise une modification du modèle suivant les messages qu'il a reçu et transmet une nouvelle version mise à jour du modèle aux nœuds de calcul. La stratégie employée par le Parameter Server pour mettre à jour le modèle global peut varier : le serveur peut attendre les messages de tous les workers (pour en prendre la moyenne

[5] https://users.soe.ucsc.edu/~niejiazhong/slides/chandra.pdf

par exemple), ou le serveur peut réaliser des modifications asynchrones en fonction de ses messages (Li M., et al. 2013). Cette méthode est particulièrement utile lorsque les modèles sont très complexes comme les réseaux de neurones profonds.

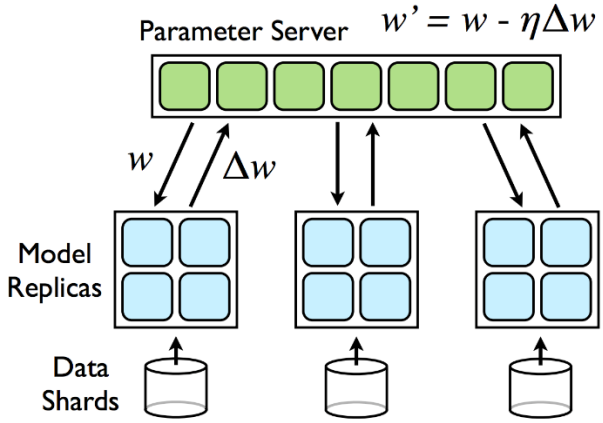

Figure 53: Schéma expliquant le fonctionnement d'un parameter server pour l'optimisation. © Tensorflow

4.2.4 Discussions

Les méthodes précédentes sont comparées suivant 3 critères de comparaison :

- Passage à l'échelle : c'est l'évolution du temps de calcul d'une méthode en fonction de la taille de la base de données en entrée. Deux propriétés sont étudiées en particulier : la « strong scalability » qui est l'évolution du temps de calcul en fonction de nombre de cœurs à charge de travail fixe et la « weak scalability » qui est l'évolution du temps de calcul en fonction de nombre de cœurs à charge de travail fixe par cœur.
- Robustesse à la répartition des données : est-ce que la répartition des données a un impact sur les performances de la méthode ? Les partitions de données ne sont pas nécessairement aléatoires et les données peuvent être regroupées par similarité.
- Performance globale de la méthode : niveau de performance de la méthode distribuée par rapport à l'état de l'art local.
- La complexité en termes d'opérations et de communications de la méthode

Nom de la méthode	Quantité distribuée		Principe
	Données	Paramètres	
Best CV		X	Recherche du meilleur classifieur en parallèle sur une base d'exemples communes
Planet/Xgboost	X		Boosting d'arbres de décision appris de manière distribuée par aggrégation d'histogrammes dans les feuilles
Distboost	X		Boosting de classifieurs faibles calculés par une moyenne de classifieurs appris par machine
Adasampling	X		Extraction d'exemples difficiles à classifier par machine et apprentissage local du classifieur final
Sybil	X	X	Paralélisation des calculs par exemples et classifieurs fixés par avance
Multiboosting	X	X	Moyenne de classifieurs de type boosting appris par machine
Méthode proposée	X		Boosting de classifieurs faibles calculés par la sélection du meilleur classifieur parmi un ensemble fixé

Tableau 4: Récapitulatif des méthodes de Boosting distribué

Nom de la méthode	Quantité distribuée		Passage à l'échelle		Facteur de communication	Mises à jour
	Données	Paramètres	Données	Machines		
Best CV		X	No	Yes	---	Itérative
Planet/Xgboost	X		Yes	Yes	+++	Itérative
Distboost	X		No	No	+	Itérative
Adasampling	X		No	No	---	Itérative
Sybil	X	X	Yes	Yes	++	Parallèle
Multiboosting	X	X	Yes	No	---	Itérative
Ours	X		Yes	Yes	++	Itérative

Tableau 5: Comparatif des méthodes de Boosting distribué

Les méthodes « parameter parallel » ne sont pas robustes à l'augmentation de données : en effet, les données doivent entièrement contenir en mémoire d'une machine pour leur exécution. L'analyse de scalabilité montre que les méthodes « data parallel » sont préférables car elles peuvent s'exprimer en programme de flot de données. Un flot de données possède des propriétés intéressantes de passage à l'échelle.

Les méthodes basées sur l'extraction d'exemples Adasampling et R/D-sampling sont limitées lorsque l'on passe à l'échelle. En particulier, le goulot d'étranglement se trouve dans l'apprentissage local du classifieur final qui ne tire pas parti de l'exécution distribuée. Un tel algorithme est principalement basé sur le transfert de données vers la machine centrale : l'utilisation du cluster n'est pas optimisée. De plus, les performances de cette méthode reposent principalement sur le choix de la méthode d'extraction qui doit en particulier être robuste au partitionnement des données. Le partitionnement des données n'est pas nécessairement uniforme et les données peuvent être regroupées par similarité par exemple. En particulier, les stratégies d'extraction d'exemples difficiles proposées ne sont pas robustes à un partitionnement non uniforme. De plus, les stratégies d'extraction sont limitées par le nombre maximal d'exemples que peut traiter une seule machine.

La méthode Distboost souffre d'un problème similaire. Pour rappel, la méthode est basée de l'apprentissage du classifieur faible sous la forme d'une somme de classifieurs appris sur chaque partition

de données. La méthode est alors dépendante de la répartition des données sur les machines car les classifieurs appris apprennent à différencier les exemples au sein d'une même partition et non des exemples provenant de partitions différentes. La répartition doit donc être uniforme pour que la méthode fonctionne. De plus, la méthode est très peu efficace en termes de calculs car certains classifieurs appris sur les partitions peuvent être similaires. En particulier, si les distributions sont uniformes, les classifieurs appris sur chaque partition seront les mêmes. Les communications seront alors redondantes tout comme les opérations nécessaires pour la prédiction par le classifieur final. Ces deux phénomènes sont accentués si la quantité de données augmente. La méthode Distboost ne passe pas à l'échelle pour les données sans modifications.

Parmi les méthodes « data parallel », les méthodes basées sur la construction d'arbres distribués nécessitent beaucoup de calculs et communications pour apprendre le classifieur final. Pour rappel, la méthode est basée sur la croissance d'un arbre de décision où un histogramme est calculé par feuille de l'arbre pour déterminer la séparation pour les prochaines feuilles. Cette méthode nécessite alors beaucoup d'itérations sur la base de données puisqu'il en faut autant que la profondeur de l'arbre souhaité et autant d'arbres que d'itérations de Boosting. Cette méthode requiert un trop grand nombre d'itérations sur les données.

Parmi les méthodes hybrides, l'approche Multiboosting n'a pas de garantie sur sa robustesse à la mauvaise répartition des données. Pour rappel, un classifieur est appris par partition de données et le classifieur final consiste à prendre la moyenne de ces classifieurs. L'augmentation du nombre de partitions augmente le nombre de classifieurs du classifieur final qui a pour effet d'éviter le sur-apprentissage mais pas d'augmenter les performances. En réalité, il faut que les partitions soient suffisamment grandes pour que les classifieurs appris soient performants. Les performances du classifieur final sont intimement liées aux performances des classifieurs sur leur partition respective. De la même manière que Distboost, l'approche Multiboosting peut souffrir d'une redondance potentielle des classifieurs si les partitions sont trop similaires. De plus, le Multiboosting souffre également du fait que les séparations de classes apprises ne séparent que des classes présentes dans les mêmes partitions.

La structure du Parallel Boosting a été conçue pour le passage à l'échelle. Cependant, l'idée principale est que tous les classifieurs soient ajoutés au classifieur final et leurs coefficients d'importance sont estimés au fur et à mesure de l'apprentissage. Par conséquent, la méthode n'est pas robuste au nombre de descripteurs utilisés et en particulier au nombre fixé de classifieurs faibles. En effet, l'ajout automatique de tous les classifieurs faibles dans le classifieur final supprime l'étape de sélection de classifieurs originellement présente dans le Boosting. Un nombre de classifieurs trop important peut alors nuire aux performances du classifieur final. De surcroit, cela supprime une propriété fondamentale du Boosting qui permet de sélectionner les descripteurs de manière pertinente.

Les Parameter Servers n'ont pas été appliqués à des modèles de Boosting. Le principe de l'utilisation de l'asynchronisme des mises à jour semble également aller à l'encontre de la mise à jour itérative du classifieur final comme le fait le Boosting. De plus, les Parameter Servers sont formulés pour des descentes de gradient et devraient être adaptés pour l'utilisation du Boosting.

4.3 UNE NOUVELLE METHODE DE BOOSTING DISTRIBUE

4.3.1 Aperçu général

Suite à l'analyse des méthodes distribuées existantes, des bonnes pratiques de conception ont pu être extraites et appliquées à notre cas d'utilisation. L'objectif est de construire une implantation du Boosting qui puisse passer à l'échelle : dans ce cadre-là, la construction se fera sous la forme d'un algorithme de flux de données. Un ensemble fixe de classifieurs faibles est utilisé pour un apprentissage efficace. Cet ensemble fixe de classifieurs est choisi en discrétisant les variables : l'espace continu des attributs numériques est remplacé par un espace discret. Différentes stratégies de discrétisation existent mais deux principales stratégies sont utilisées dans ces analyses : une méthode de discrétisation régulière par attribut numérique, et une méthode sur des histogrammes équirépartis (utilisation de quantiles approximés). D'autres stratégies de discrétisation ont été utilisées notamment une stratégie basée sur des apprentissages de Boosting locaux comme PreWeak. Mais, de manière générale, la formulation actuelle de la méthode proposée n'impose aucune contrainte sur l'ensemble de classifieurs. Le Boosting est composée de deux étapes fondamentales : l'apprentissage d'un classifieur faible et la mise à jour des poids. La mise à jour des poids est nativement parallélisable et les stratégies de distribution se concentrent sur la distribution de l'apprentissage de classifieurs faibles. L'apprentissage classique d'un classifieur faible est remplacé par la sélection du meilleur classifieur d'un ensemble fixe selon son erreur de généralisation.

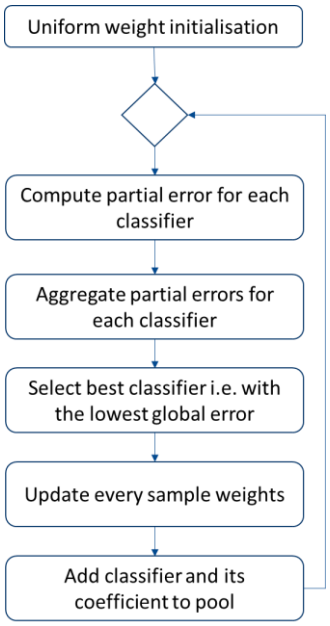

Figure 54: Schéma récapitulant le principe général mis en place pour le boosting distribué.

4.3.2 Discrétisation des variables

La méthode proposée est basée sur l'utilisation d'un ensemble fixe de classifieurs faibles et nécessite le calcul de leurs erreurs de généralisation. Les éléments de cet ensemble sont cruciaux et peuvent être construits de deux manières : soit par l'utilisation de classifieurs externes soit par la construction de classifieur faible à partir des variables. Le premier cas est suffisamment rare pour le mettre de côté pour le moment. Les stratégies étudiées ici sont des stratégies basées sur l'étude indépendante des attributs

numériques et d'autres stratégies plus avancées pourront être étudiées dans un deuxième temps. La première stratégie est la stratégie de discrétisation régulière et consiste à construire des seuils équidistants dans l'intervalle de valeurs possibles de chaque attribut numérique. Dans certains cas, la méthode se révèle peu performante (spécialement si les données ne sont pas réparties uniformément sur l'intervalle) mais très peu coûteuse en termes de calcul. L'utilisation de quantiles équirépartis (Chen T. et Guestrin C. 2016) (Panda B., et al. 2009) permet de calculer une meilleure discrétisation : ces quantiles peuvent être estimés de manière distribuée ou sur un sous échantillon. Des méthodes comme PreWeak utilisent les seuils utilisés dans les classifieurs faibles des ensembles de Boosting appris localement : les modèles locaux sont appris sur des régions homogènes des données locales pour avoir une meilleure discrétisation. Il est important d'avoir une discrétisation non redondante de manière à ne pas avoir de calculs redondants ou sans information lors de l'exécution de l'algorithme. De plus, un ensemble constitué de 2 classifieurs opposés est suffisant car le Boosting est capable d'inverser la décision d'un classifieur grâce au paramètre α_t. Dans le cas des arbres à 1 niveau, les deux sens de décision ne doivent donc pas être considérés et un seul est suffisant dans l'ensemble des classifieurs.

Dans le cadre de cette étude, les arbres de décision ont été choisis et la discrétisation se construit de la manière suivante. On rappelle que $h_{s,\rho,p}(\boldsymbol{X}) = \mathbb{I}[p\boldsymbol{X}^s < p\rho]$. Un seul sens de comparaison est étudié : $p = +1$. $s \in \{1, \dots, d\}$ car tous les attributs numériques sont considérés ($\boldsymbol{X} \in \mathbb{R}^{\mathrm{d}}$). Pour une formulation générale, on note Γ^s l'ensemble des seuils utilisés pour l'attribut s, c'est-à-dire :

$$\mathcal{H} = \left(h_{\rho,s,+1}\right), s \in \{1 \dots d\}, \rho \in \Gamma^s.$$

L'hypothèse simplificatrice que tous les attributs sont normalisés entre 0 et 1 peut être faite sans perte de généralité. La stratégie de discrétisation ici utilisée pour les expériences est la stratégie de discrétisation régulière où $\rho \in \left\{\frac{1}{N_d}, \frac{2}{N_d}, \dots, \frac{N_d-1}{N_d}\right\}$ et N_d est le nombre de seuils considérés. Alors :

$$\mathcal{H} = \left(h_{\rho,s,+1}\right), s \in \{1, \dots, d\}, \rho \in \left\{\frac{1}{N_d}, \frac{2}{N_d}, \dots, \frac{N_d - 1}{N_d}\right\}$$

4.3.3 Calcul des erreurs

Pour sélectionner le meilleur classifieur, les erreurs de généralisation sont calculées pour chaque classifieur. L'erreur de généralisation a l'avantage d'être une grandeur additive et peut être calculée en sommant les contributions de chaque partition. Une première opération exécutée en parallèle sur tous les nœuds de calcul (Map) consiste à calculer l'erreur de généralisation locale et la deuxième opération consiste à calculer la somme des erreurs pondérées par la population de la partition sur toutes les partitions grâce à une opération Reduce qui calcule la somme pondérée de deux éléments (voir Tableau 6 pour plus d'informations). Supposons que l'ensemble des données S est divisé (partitionné) en K partitions $S = \left(S^k\right), k = 1, \dots, K$, le calcul décrit se résume à :

$$\epsilon(S, h, D_t) = \sum_{machine} error_h(machine) = \sum_{k=1}^{K} \epsilon\left(h, S^k, D_t^k\right)$$

FONCTION Map_Partition(S^k, h, D_t^k) :
- Calcule et renvoie $\epsilon\left(h, S^k, D_t^k\right)$

FONCTION Reduce_Partition($\epsilon\left(h, S^k, D_t^k\right)$, $\epsilon\left(h, S^{k'}, D_t^{k'}\right)$)
- Retourne $\epsilon\left(h, S^k, D_t^k\right) + \epsilon\left(h, S^{k'}, D_t^{k'}\right)$

Tableau 6: Fonctions utilisées pour la sélection distribuée.

4.3.4 Sélection du meilleur classifieur

Une fois que l'erreur globale est calculée pour tous les classifieurs faibles, elle est utilisée afin de sélectionner le meilleur classifieur :

$$h_i = best\left(\sum_{machine} error_h(machine) \quad , h \in \mathcal{H}\right)$$

Le classifieur sélectionné est le classifieur qui a la plus petite erreur. Cependant, seulement la moitié des classifieurs est considérée à ce moment là puisque les classifieurs opposés sont incorporés avec l'inversion du signe de α_t. Le critère de sélection doit prendre en compte les performances du classifieur opposé : pour cela, la frontière (edge) d'un classifieur a été introduit $\epsilon(S, h, D) = 0.5 - \gamma(S, h, D)$ et sa valeur absolue est utilisée comme critère de sélection. Les frontières d'un classifieur et de son conjugué sont opposées.

For $k = 1, \ldots, K$ (on each machine) (Map)
- Compute $\epsilon\left(S^k, h, D_t^k\right), h \in \mathcal{H}$

$\forall h \in \mathcal{H}$, $\epsilon(S, h, D_t) = \sum_{k=1}^{K} \epsilon\left(S^k, h, D_t^k\right)$ (Reduce)

$h^* = \underset{h \in \mathcal{H}}{\operatorname{argmax}} |\gamma(S, h, D_t)|$

Algorithme 4: Apprentissage d'un arbre de décision à un niveau de manière distribuée.

4.3.5 Conclusions

Toutes les briques décrites précédemment peuvent être assemblées pour construire l'algorithme final de Boosting distribué basé sur un ensemble fixe de classifieurs. Cet algorithme a été proposé dans (Le Goff M., Tourneret J-Y et Wendt H., et al. 2016).

Inputs:

- Base de données d'exemples partitionnés $S = (\boldsymbol{x}_i, y_i)_{i=1,..,N} = \cup_{k=1,..,K} S^k$
- T le nombre d'itérations du Boosting
- $\mathcal{H}$ un ensemble de classifieurs faibles

$t \leftarrow 0$

$\forall\, i = 1 \ldots N, w_i^0 = \frac{1}{N}$

While t < T

// Calcul des erreurs partielles pour chaque machine en parallèle

For $k = 1, \ldots, K$ (on each machine) (Map)

// Echantillonnage préférentiel

- Génération de $\left(\tilde{S}^k, \tilde{D}_t^k\right)$ par échantillonnage préférentiel de $\left(S^k, D_t^k\right)$

// Calcul des erreurs partielles pour chaque classifieur

- Compute $\epsilon\left(\tilde{S}^k, h, \tilde{D}_t^k\right), h \in \mathcal{H}$

EndFor

// Calcul de l'erreur totale en sommant toutes les erreurs partielles

$\forall h \in \mathcal{H}\ , \epsilon\left(\tilde{S}, h, \tilde{D}_t\right) = \sum_{k=1}^{K} \epsilon\left(\tilde{S}^k, h, \tilde{D}_t^k\right)$ (Reduce)

// Sélection du meilleur classifieur en fonction de leur frontière

$$h^* = \underset{h \in \mathcal{H}}{\operatorname{argmax}} \left|\gamma\left(\tilde{S}, h, \tilde{D}_t\right)\right|$$

// Calcul du coefficient d'importance du classifieur

$$\alpha_t = 0.5 \log \frac{1 - \epsilon_{pratique}^t\ (\tilde{S}, h_t, \tilde{D}_t)}{\epsilon_{pratique}^t\ (\tilde{S}, h_t, \tilde{D}_t)}$$

// Calcul de la constante de normalisation

$Z_{t+1} = \sum_{i=1 \ldots N} w_i^t\ \exp(-\alpha_t y_i\ h_t(x_i))$ (Reduce)

// Mise à jour des poids selon le principe du Boosting

$\forall\, i = 1 \ldots N, \mathrm{w_i^{t+1}} = \frac{\mathrm{w_i^t}}{Z_{t+1}} \exp(-\alpha_t y_i\ h_t(x_i))$ (Map)

endWhile

Algorithme 5: Algorithme proposé pour un modèle de Boosting distribué.

L'avantage principal de la méthode est son efficacité en termes de calcul. Toutes les étapes sont exécutées en parallèle contrairement à d'autres méthodes. Par conséquence, la méthode possède une complexité linéaire et passe à l'échelle. Les différentes modifications permettent d'augmenter la complexité et la robustesse du classifieur final.

4.4 VALIDATION ET EXPERIMENTATIONS

4.4.1 Description des scenarios de simulation

Nous proposons d'évaluer notre méthode de Boosting sur le problème de la détection de nuages à partir de l'analyse d'une base de données SPOT 6. La base de données sur laquelle a été testé cet algorithme est composée d'images album SPOT 6 qui possèdent quatre canaux spectraux : le rouge, le vert, le bleu, et le proche infrarouge. La résolution spatiale des images album est de 60m. En effet, les images album sont obtenues en dégradant des images SPOT 6 pleine résolution dont la résolution initiale est de 1.5 mètres. Les composantes spectrales sont quant à elles encodées sur 12 bits. Chaque image est associée à son masque de nuages qui a été déterminé par des opérateurs humains. La base de données à disposition contient plus de 100000 images couvrant la surface terrestre. Notons que ces images offrent une grande diversité de paysages et de nuages. La base de données a été mise à disposition par Airbus Defence And Space et les images sont corrigées géométriquement et radiométriquement. Notons que la taille de la base de données est adéquate pour évaluer le passage à l'échelle de l'algorithme.

Des descripteurs numériques sont calculés à partir des images afin de les utiliser comme entrée pour la méthode du Boosting. Plusieurs familles de descripteurs ont déjà été utilisées avec succès pour cette application de détection de nuages (Chandran A. et Christy J. 2015). Pour notre cas d'utilisation, les ratios entre bandes (quotient entre les intensités associées à deux bandes spectrales) sont utilisés car ce sont des descripteurs par pixel et connus pour leur performance (Hollstein A., et al. 2016). L'utilisation des quotients de bandes a déjà montré des performances intéressantes sur des cas d'utilisation comme la détection de nuages (Hollstein A., et al. 2016). Ces descripteurs ont l'avantage d'être simples à calculer et indépendants des conditions d'illumination. D'autres familles de descripteurs sont étudiées pour comparer les performances : les coefficients issus d'une décomposition en ondelettes ont été souvent utilisés pour faire de la reconnaissance d'images (Noureldin L., et al. 2012) et sont utilisés à titre de comparaison. En particulier, nous considérons deux décompositions en ondelettes (Gabor et Discrete Cosine Transform) qui ont déjà été utilisées pour faire de la détection de nuages (Noureldin L., et al. 2012). De même, différentes résolutions spatiales (60m, 120m and 240m) sont utilisées pour calculer les ratios et ainsi améliorer les performances avec une approche multi échelle. Des descripteurs supplémentaires pourront être rajoutés par la suite comme l'utilisation d'indices tels que le NDCI (Normalized Difference Cloud Index) ou le NDSI (Normalized Difference Snow Index) (Hollstein A., et al. 2016).

Les expériences à mener servent à déterminer les propriétés de passage à l'échelle de l'algorithme et ses performances finales. Ces expériences sont basées sur l'utilisation du cloud en particulier du Google Cloud qui permet de louer les ressources de calcul nécessaires. Cette flexibilité est utilisée pour exécuter un apprentissage en variant la configuration du cluster. Une configuration classique de cluster est composée de 4 machines de 8 cœurs de calcul et 30Go de mémoire et le nombre de machines est augmenté pour de plus gros clusters (de 4 à 32, par exemple). Pour information, le prix correspondant à la location d'une telle machine est de 0.33€ par heure d'utilisation. De la même manière, la taille de la base de données

d'entrainement est modifiée entre scénarios : les bases de données utilisées comportent 500, 1000, 5000 et 10000 images différentes. Les tailles de ces bases de données à traiter sont de 7Go, 15Go, 70Go, et 150Go. A noter, ces tailles correspondent aux tailles des bases de données brutes avant le calcul des descripteurs : le calcul des descripteurs augmente la taille par un facteur multiplicatif proportionnel au nombre de descripteurs utilisés. Comme présentés précédemment, les descripteurs utilisés sont les ratios entre bandes à 3 échelles ($d = 18$ descripteurs différents), les coefficients de Gabor pour 4 rotations et 3 échelles ($d = 48$) et les coefficients de la Discrete Cosine Transform ($d = 48$). A titre indicatif, la base de données la plus importante possède une taille excédant les 220Go. La taille minimale de 500 images a été conçue afin de représenter un traitement classique d'apprentissage non distribué : cette taille a été choisie en fonction de la mémoire disponible sur une machine quelconque. Les scénarios d'expériences consistent à mesurer le temps d'apprentissage pour chaque taille de base de données et chaque configuration de cluster. L'étude de ces temps d'apprentissage permettra d'établir les propriétés essentielles de l'algorithme proposé vis-à-vis du passage à l'échelle.

Le deuxième type d'expériences a pour but de vérifier la qualité des performances des classifieurs obtenus avec ce nouvel algorithme. La méthode distribuée est d'abord comparée à la méthode classique sur un scénario d'exécution classique de la méthode locale. Un même jeu de données est utilisé pour comparer l'évolution de l'erreur d'apprentissage au fur et à mesure des itérations de la procédure de Boosting. Le jeu de données utilisé pour cette expérience est la plus petite base de données pour convenir à la méthode locale et sur laquelle les ratios de bandes ont été calculés. La seconde propriété à être surveillée est l'évolution de l'erreur de généralisation en fonction de la taille de la base de données en apprentissage. Différents jeux de données sont générés : de tailles variables pour l'apprentissage et un dernier contenant 500 images non incluses dans les échantillons d'apprentissage qui servira à comparer les méthodes. Classiquement, le pourcentage de bonne classification sera utilisé comme métrique de comparaison et les scores sur la base d'entrainement et sur la base de test seront utilisés pour départager les classifieurs. De la même manière, des classifieurs utilisant différentes familles de descripteurs seront comparés : une version pour chaque famille sera apprise sur une base de 1000 images pour les comparer sur une base de test.

Et enfin, le dernier axe de comparaison est la comparaison qualitative des performances visuelles des classifieurs. Les prédictions issues des différents classifieurs sont utilisées pour générer des masques de nuages qui sont ensuite comparés aux masques réels. Les images sont divisées en plusieurs catégories dépendant de leur couverture nuageuse (beaucoup, peu et pas de nuages) et les performances sont évaluées par catégorie. L'objectif de cette dernière analyse est de repérer les artefacts de mauvaise classification afin d'identifier des pistes d'amélioration.

4.4.2 Passage à l'échelle

Les expériences suivantes sont menées afin de vérifier l'efficacité du passage à l'échelle de notre algorithme de Boosting. Pour cela, les expériences consistent à faire varier les paramètres affectés par le passage à l'échelle :

- Moyens de calcul mis en place matérialisés par le nombre de cœurs réquisitionnés par le cluster de calcul
- Taille du problème matérialisée par la taille de la base de données

Le passage à l'échelle est mesuré par le fait que le temps de calcul évolue linéairement avec la quantité de données. On mesure le passage à l'échelle d'une méthode grâce à l'efficacité du système qui est le rapport entre la taille du problème et les moyens mis en place pour le résoudre : le nombre d'exemples et le temps CPU utilisé pour le résoudre. La propriété de scalabilité faible implique que l'efficacité du système doit être constante ; de cette manière, si la taille du problème augmente, il est suffisant d'augmenter la puissance de calcul du même facteur pour maintenir le temps de calcul constant. La Figure 55 montre que l'efficacité du système reste identique pour différentes tailles de problèmes et de cluster. L'efficacité dépend du nombre de variables à traiter : en effet, il n'y a que 18 ratios à traiter et une cinquantaine pour les autres familles. Afin d'avoir une idée de la taille des données traitées, la taille de la base de données de 500 images est de 5 Go pour les descripteurs du type Ratios et de 10 Go pour les autres descripteurs. Et de la même manière, la base de données contenant les 1000 images a une taille de 10 Go pour les ratios et de 25 Go pour les autres descripteurs. La base de données de 5000 images justifie largement l'utilisation de techniques distribuées puisqu'elle a une taille de 45 Go pour les ratios et 110 Go pour les autres descripteurs (1000 images : 80 Go pour les ratios et 220Go pour les autres).

La Figure 55 montre l'efficacité de la méthode pour les différentes tailles de cluster : l'efficacité semble être indépendante de la taille du cluster d'exécution dans le cadre des expériences menées. Des tailles de bases de données adéquates ont été utilisées pour limiter l'impact de l'initialisation et de communications initiales sur l'estimation de l'efficacité. Le type de descripteurs a un impact important sur l'efficacité. Ce résultat était attendu car le nombre de descripteurs (et alors de seuils à considérer dans l'algorithme) change suivant leur type (de 18 à 48). A titre d'exemple, l'analyse d'une base de données de 1000 images sur un cluster de 4 machines 8 cœurs (soit $4 \times 8 = 32$ cœurs) requiert 40 minutes d'exécution.

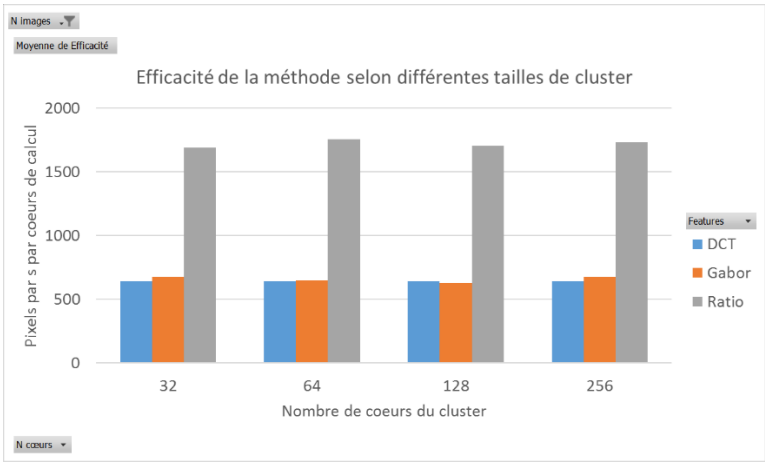

Figure 55: Efficacité de l'algorithme sur différentes configurations de cluster.

La deuxième expérience (Figure 56) consistait à varier la taille de la base de données pour mesurer l'efficacité de l'algorithme pour des bases de données de grande taille. De la même manière que précédemment, pour une grande partie des bases de données, l'efficacité est constante, ce qui confirme le passage à l'échelle de l'algorithme. Cependant, on peut noter une baisse de l'efficacité pour les petites tailles de bases de données : les phénomènes d'initialisation perturbent beaucoup l'estimation de l'efficacité pour des temps trop faibles de calcul.

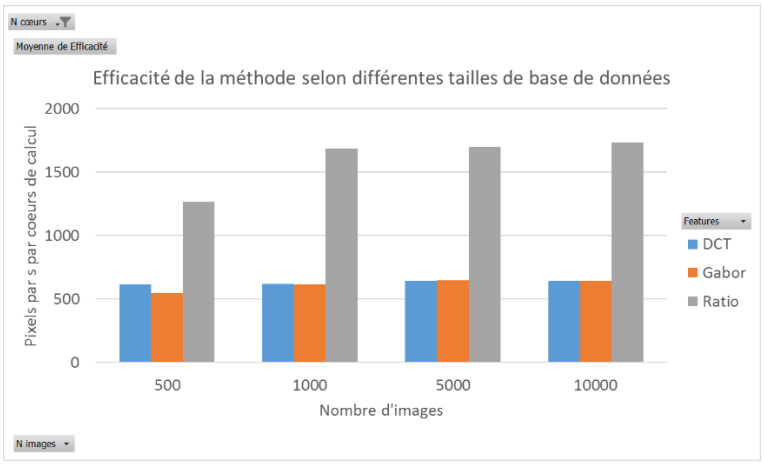

Figure 56: Efficacité de la méthode pour différentes tailles de bases de données.

La seconde propriété à évaluer est la propriété dite de passage à l'échelle fort : la taille du problème est fixe et le nombre de cœurs du cluster est augmenté. Cette expérience permet de répondre à la question de l'accélération du temps de calcul en augmentant la puissance de calcul mise en place. Une relation linéaire impliquerait une décroissance linéaire du temps de calcul. Cependant, le nombre de cœurs ne peut pas être trop augmenté car les effets de communications deviennent importants relativement à leur charge de travail. Cependant, sur un intervalle raisonnable de travail, la relation linéaire et le passage à l'échelle peuvent être observés (voir Figure 57).

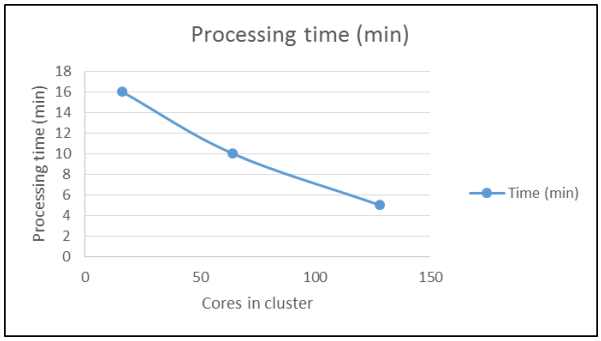

Figure 57: Passage à l'échelle fort où à taille de problème fixe, le nombre de cœurs du cluster est augmenté

4.4.3 Performance de classification

Les performances des différentes méthodes d'apprentissage doivent être évaluées, tout d'abord d'un point de vue quantitatif avec le calcul des performances sur une base de test et d'entrainement et d'un point de vue qualitatif en étudiant la morphologie des prédictions. Les performances sont évaluées par le biais de l'erreur de classification sur la base qui a servi pour l'entrainement et sur une autre base externe dite de test. L'erreur de classification est calculée par le rapport du nombre de fois où le classifieur a prédit la bonne quantité sur le nombre total d'exemples.

De manière simple, les performances observées sont similaires à celle attendues par la comparaison à l'état de l'art local. La première expérience consistait à vérifier si l'algorithme de Boosting proposé convergeait de la même manière que la version classique. La Figure 58 montre que la convergence est similaire avec un avantage pour la méthode locale qui bénéficie du fait qu'aucune hypothèse de simplification n'ait été faite. La discrétisation des features simplifie le problème et diminue légèrement les performances. La performance de la méthode locale est meilleure pour des petites quantités de données. Cependant, l'augmentation de données est supposée aider à rattraper ce déficit de performances.

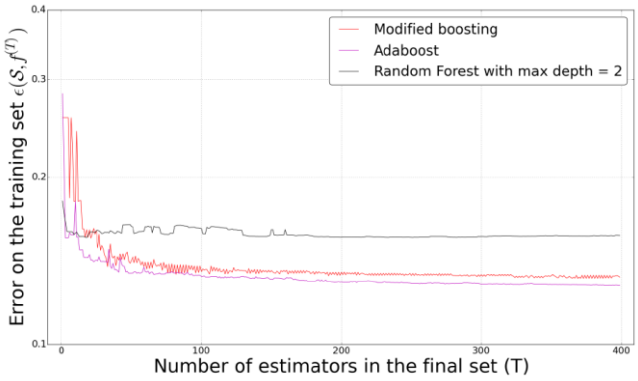

Figure 58: Comparaison des erreurs de classification pendant l'entrainement des deux méthodes (locale et distribuée).

Naturellement, les performances de généralisation en fonction de la quantité de données d'entrainement ont été également mesurées et reportées sur la Figure 59. A jeu de données égal, la méthode locale est plus performante à la fois sur la base de données d'entrainement et sur la base de données de test que la méthode distribuée. L'erreur sur la base d'entrainement de la méthode distribuée augmente lorsque la taille de la base d'entrainement augmente car le problème d'apprentissage augmente en complexité avec le nombre d'exemples. Cependant, l'erreur de généralisation diminue avec le nombre d'exemples, ce qui encourage l'utilisation de plus de données. Il faut noter que la méthode locale ne peut pas analyser une base de données de plus de 500 images. Il faut cependant remarquer que l'apport entre l'analyse de 1000 images et 5000 images reste faible comparativement à la quantité de calcul nécessaire (le facteur d'échelle est de 5).

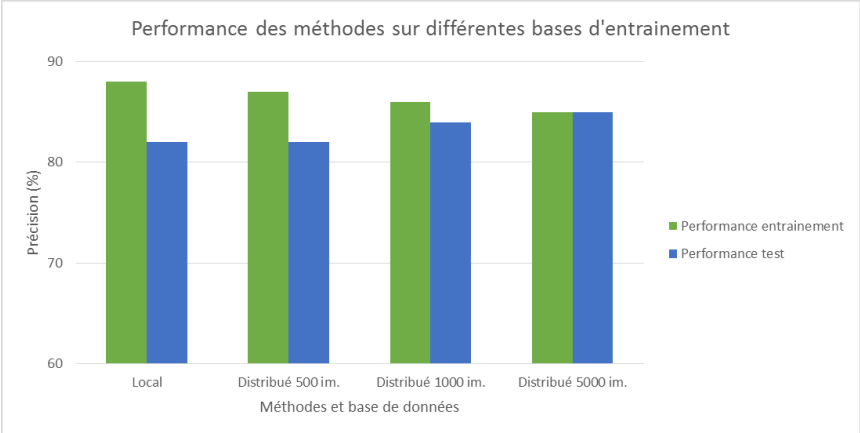

Figure 59: Performances sur les ensembles d'entrainement et de test des différents algorithmes. La précision est l'opposé de l'erreur de classification.

Les prochains tests consistent à analyser l'utilisation de plusieurs jeux de descripteurs : les coefficients issus d'une décomposition de Gabor, les coefficients calculés à partir d'une décomposition en cosinus discrets (DCT), et les ratios. Un classifieur a été appris pour chaque jeu de descripteurs sur 1000 images et leurs performances ont été comparées sur une base d'images externes. Les ratios semblent avoir les meilleures performances (Figure 60) mais la décomposition en ondelettes de Gabor possède des performances similaires quoique légèrement inférieures.

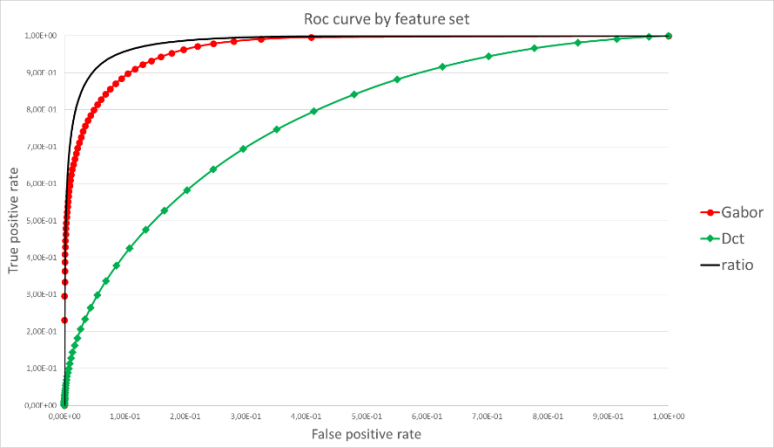

Figure 60: Courbe ROC des différents ensembles de descripteurs appris par le classifieur distribué

De la même manière, la dernière phase de comparaison consistait à analyser les prédictions du classifieur sur des images entières. Cela pose la question de la répercussion de l'erreur statistique mesurée sur une base de données constituée de véritables images. Les images sont séparées en plusieurs catégories suivant leur couverture nuageuse : en réalité, ces catégories sont liées à la difficulté de traiter une image pour le classifieur. Les images contenant beaucoup de nuages comme celles sur la Figure 61 sont en réalité faciles à classifier car certains nuages ont des signatures très caractéristiques (gros, blanc et épais).

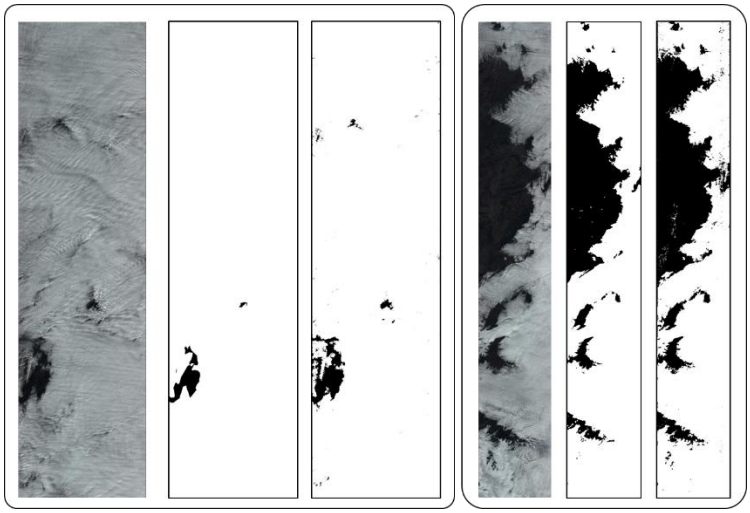

Figure 61: Exemples de classification sur des images contenant beaucoup de nuages. Gauche : dans l'ordre, image initiale contenant presque exclusivement des nuages, masque de nuages et masque prédit par le classifieur. Droite : dans l'ordre, image initiale contenant en grande partie de nuages, masque de nuages et masque prédit par le classifieur.

Le deuxième type d'images sont des images contenant quelques nuages comme l'illustre la Figure 62. Les performances de classification pour cette catégorie sont plus variables mais restent convenables comme le montre Figure 62. Cependant, ces résultats permettent déjà de mettre en avant des faiblesses de l'approche de classification pixel par pixel. Les masques de nuages contiennent des formes plus faciles à classifier que les pixels isolés. La classification par pixel induit des erreurs de classification isolées qui statistiquement sont négligeables mais visuellement sont importantes. Cette remarque implique l'utilisation de techniques de traitement d'images ou de prédiction structurée pour « lisser » les prédictions. Cette amélioration sera un bon axe de développement de notre approche de Boosting mais n'est pas considérée (par manque de temps) dans le cadre de cette étude.

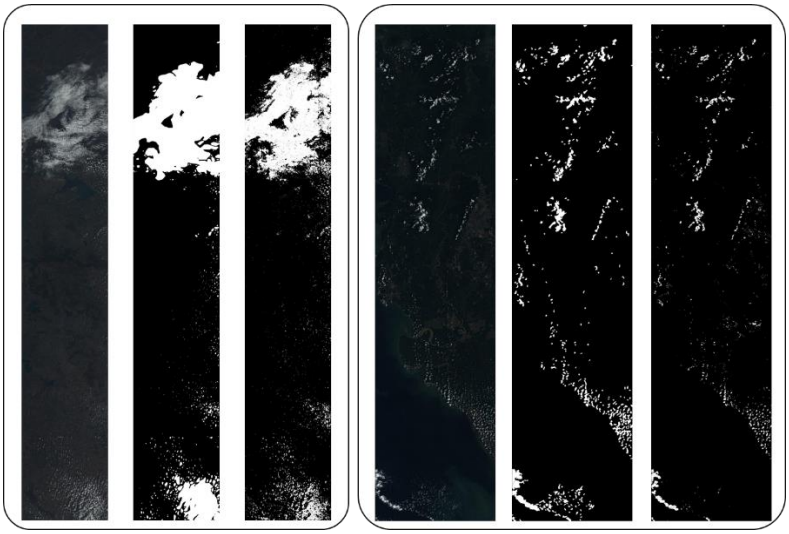

Figure 62: Exemples de classification d'images contenant des nuages. Gauche : dans l'ordre, image initiale, masque de nuages et masque prédit par le classifieur. Droite : dans l'ordre, image initiale d'un paysage côtier, masque de nuages et masque prédit par le classifieur.

Les erreurs de classification se retrouvent majoritairement dans la dernière catégorie d'images contenant peu ou pas de nuages qui contiennent, en particulier, des objets spectralement similaires à des nuages : comme des sables côtiers, ou de le neige (voir Figure 63). Les objets que l'on peut observer sur la Figure 63 sont la principale source d'erreur de classification. Ce phénomène est expliqué à priori par la simplicité des descripteurs numériques utilisés. De la même manière, une faiblesse de la détection de petits nuages peut être mis en évidence (Figure 63).

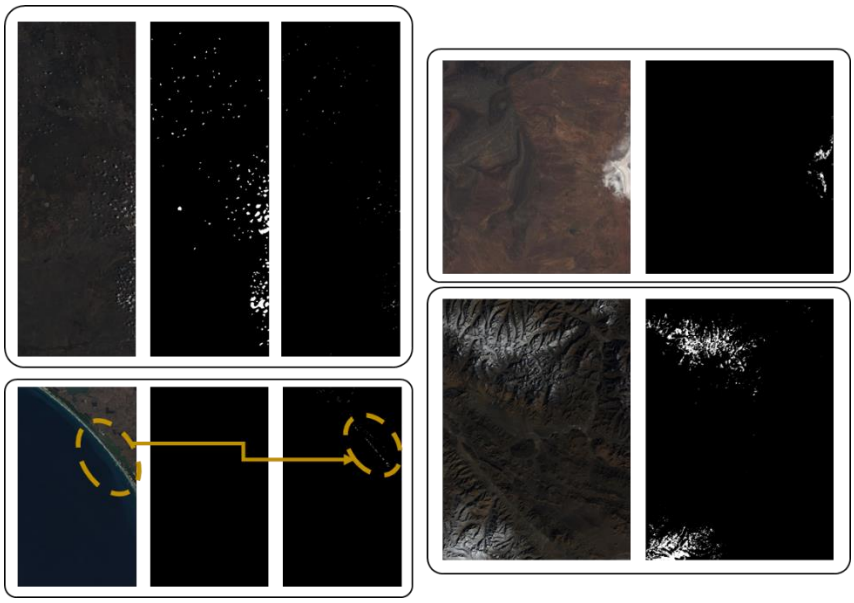

Figure 63: Exemples d'artefacts de mauvaises classifications.

Au-delà des performances numériques, de réelles différences de performance peuvent être mises en évidence lors de l'utilisation des différents types de descripteurs (Figure 64). Il semblerait que les descripteurs basés sur des ondelettes ne possèdent pas la même continuité spatiale que les descripteurs basés sur des ratios, ce qui cause des artefacts de mauvaise classification. En réalité, cela pose la question fondamentale : la forme de la décision contenant une succession de seuils n'est peut-être pas adaptée à l'utilisation d'une décomposition en ondelettes ?

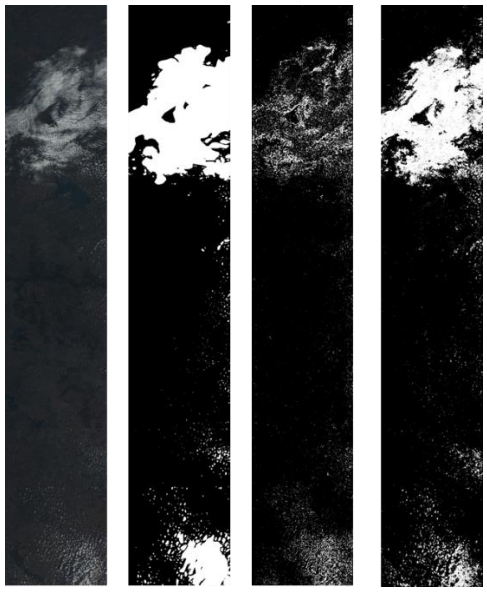

Figure 64: Comparaison de classifications entre les différents jeux de descripteurs. (De gauche à droite : Image, Masque de nuages, Prédiction utilisant les descripteurs de Gabor et Prédiction utilisant les ratios)

4.5 BILAN SUR LE BOOSTING DISTRIBUÉ

4.5.1 Conclusion

Tout d'abord, l'algorithme proposé doit être comparé aux différentes méthodes existantes dans la littérature. Le Tableau 4 résume les principales méthodes d'implantation de Boosting distribué décrites dans ce chapitre. Le Tableau 5, quant à lui, permet de présenter les principaux points de différence des différentes méthodes. Dans ce tableau, le facteur de passage à l'échelle décrit l'effet de l'augmentation de la quantité de données ou de machines sur le classifieur final et ses performances. Si le classifieur passe à l'échelle, les performances sont stables ou améliorées avec l'augmentation de l'une des deux quantités. En particulier, comme il a déjà été soulevé, certaines méthodes ne supportent pas l'augmentation de des données ou du nombre de machines. Le second critère de comparaison est le facteur de communication qui permet de noter le besoin en communication entre machines des méthodes. En particulier, ce critère est important pour des méthodes qui nécessitent un nombre important d'itérations sur les données comme l'apprentissage d'arbres dans Planet et Xgboost. Enfin, pour conclure, le dernier critère concerne la mise à jour des poids. Il permet de différencier la technique utilisée pour Parallel Boosting où tous les coefficients sont mis à jour en même temps. Une mise à jour itérative permet de conserver toutes les propriétés du Boosting incluant en particulier, la sélection de features et la sélection des exemples.

En fin de compte, la méthodologie proposée fait partie des méthodes les plus efficaces en termes de calcul et de passage à l'échelle. En particulier, l'augmentation des données et du nombre de machines augmentent les performances. De plus, la formulation de la méthode permet de conserver mes principales caractéristiques du Boosting en maintenant une mise à jour itérative des poids.

A la suite des expériences, Les conclusions tirées sont de deux types : des conclusions sur la performance de l'implantation distribuée et des performances des classifieurs appris. Les conclusions portent sur les modifications réalisées pour l'implantation distribuée mais également sur la méthodologie du Boosting choisie.

En ce qui concerne la partie informatique, il est important de rappeler quelques notions : tout d'abord, la loi d'Amdahl donne l'accélération théorique de l'exécution d'une tâche à charge de travail constante que l'on peut attendre d'un système dont on améliore les ressources :

$$\frac{1}{\alpha + \frac{1-\alpha}{N}}$$

où α est la proportion séquentielle du programme et N le nombre de cœurs du cluster par exemple. Cette formule met en évidence que le facteur limitant de l'accélération reste la partie séquentielle du programme à savoir $\alpha \in [0,1]$. Plus α est proche de 0 c'est-à-dire que la partie séquentielle est presque inexistante, plus l'accélération est importante. A l'inverse, plus la partie séquentielle est importante i.e. α proche de 1, moins l'accélération est élevée.

Les expériences de passage à l'échelle ont montré que l'algorithme est scalable car il permet d'absorber des quantités importantes de données tout en maintenant une efficacité constante. Pour augmenter encore plus son facteur de passage à l'échelle, des modifications importantes seraient nécessaires car le passage à l'échelle est limité par les opérations séquentielles nécessaires à l'application du Boosting. Ces opérations qui induisent des synchronisations régulières ralentissent l'exécution mais elles représentent également des opérations essentielles de la méthodologie du Boosting.

En ce qui concerne les performances des classifieurs, le classifieur reproduit le comportement du classifieur local et le surpasse grâce à la quantité de données qu'il a pu voir. Cependant, une conclusion classique peut être tirée des images de prédiction : les descripteurs doivent être améliorés pour améliorer les performances. Les classifieurs sont soit trop simples ou pas assez nombreux. Cependant, il est important d'étudier si la méthode est robuste à l'augmentation du nombre de descripteurs. Ses propriétés de passage à l'échelle sont connues pour les données. Une question sous-jacente à cette étude était si la quantité de données améliorait les performances. La réponse est donc positive jusqu'à un certain point puisque les performances ont stagné après 1000 images. Il faut que la complexité du classifieur appris augmente également, ce qui, dans le cas présent, faisait défaut.

4.5.2 Perspectives

Les conclusions précédentes ont permis de mettre en évidence des axes d'amélioration. D'autres variantes du Boosting ont montré de meilleures performances qu'Adaboost et une meilleure flexibilité comme la Gradient Boosting. La question de leur intégration dans la structure actuelle est intéressante à étudier. De même, l'heuristique de discrétisation des variables doit être améliorée : une modélisation plus fine des données en entrée permettrait d'avoir une discrétisation plus intelligente. Cette discrétisation est importante car le dernier axe mis en évidence par les performances est la nécessité d'accroître la complexité du classifieur en considérant plus de classifieurs faibles. Et enfin pour finir, des expériences de passage à l'échelle doivent encore être mises en place pour des tailles de données plus importantes et qui seraient plus représentatives de l'utilisation de la méthode en production et des tailles de cluster plus grandes associées.

Apprentissage profond pour la télédétection

Chapitre 5 APPRENTISSAGE PROFOND POUR LA TELEDETECTION

5.1 PROBLÈME

La partie précédente a permis de mettre en évidence que les problèmes de classification considérés nécessitent des méthodes d'apprentissage plus complexe pour atteindre de meilleurs niveaux de performance. En particulier, la quantité de données associée à une puissance de calcul adaptée n'a permis d'atteindre les performances escomptées. En réalité, de manière usuelle lors de problème d'apprentissage automatique, le choix des descripteurs est critique et souvent un facteur limitant pour les performances finales d'un système. Les techniques d'apprentissage profond ont été développées afin d'incorporer les descripteurs dans l'apprentissage et sont naturellement étudiées par la suite.

A titre de rappel, le problème à résoudre est un problème de classification d'images classique : attribuer une classe d'appartenance pour chaque pixel de l'image. Les classes considérées ici sont par exemple, la classe « nuage » et « non nuage » ou les différentes classes de terrains de la Figure 16. Or, à l'heure actuelle, l'apprentissage profond a montré de meilleures performances que les techniques historiques sur la reconnaissance d'images (He K., et al. 2016) (Krizhevsky A., Sutskever I. et Hinton G. 2012) (LeCun Y., et al. 1998).

Les techniques historiques consistent à calculer des descripteurs sur le voisinage du pixel et utiliser des techniques d'apprentissage pour apprendre le lien entre les descripteurs calculés et les classes sémantiques possibles. Il est alors nécessaire de construire des descripteurs spécifiques aux classes à reconnaître. La recherche des descripteurs optimaux est rapidement devenue un sujet de recherche important et par la suite, l'estimation des descripteurs a été ajouté à la chaîne d'apprentissage (Bengio Y., Courville A. et Vincent P. 2013). En particulier, les limites de l'utilisation de descripteurs fixes (non appris) ont été mises en évidence dans la partie précédente. L'apprentissage profond permet d'apprendre les descripteurs conjointement à la fonction de prédiction.L'apprentissage profond et en particulier, l'utilisation de réseaux de neurones convolutionnels, clé de voûte du domaine, a d'abord été formalisé dans les années 90 (Fukushima 1988). Malheureusement, le manque de puissance de calcul et de données de l'époque ne permettait pas d'utiliser ce type de technique. Comme il a été décrit dans l'état de l'art, ces techniques ont connu une croissance exponentielle ces dernières années avec les premières réalisations impressionnantes de ce genre de techniques datant de 2012 (Krizhevsky A., Sutskever I. et Hinton G. 2012). Cette évolution a été rapidement suivie par le développement et la diffusion des outils associés comme les libraires de calcul telles que Tensorflow (Abadi M., et al. 2016). Aujourd'hui, les outils et la puissance de calcul sont disponibles et il est alors crucial d'évaluer ce type de technologies sur nos cas d'utilisation.

Deux cas d'utilisation seront évalués dans cette partie. La détection de nuages est le principal cas d'utilisation et a déjà été expliquée dans le chapitre précédent. L'apprentissage profond est supposé permettre d'apprendre de meilleurs descripteurs pour détecter les nuages. Le deuxième cas d'utilisation est la classification des sols. La formulation du problème est la même que celle de la détection de nuages : un masque des sols et de nuages est associé avec l'image et doit être prédit. Le problème est similaire à l'exception du nombre de classes qui est égal à 2 pour la détection de nuages est égal à 18 pour la classification des sols (voir Figure 16). De plus, peu de connaissances sont disponibles sur le sujet puisque le problème tel qu'il est posé est nouveau : la combinaison de l'utilisation d'images album SPOT 6 et de la

vérité terrain USGSS (Broxton P. 2014) est nouvelle. Ainsi, aucun descripteur n'est particulièrement indiqué pour ce problème et l'apprentissage profond permet d'éviter de concevoir de complexes descripteurs pour cette utilisation.

5.2 MODÉLISATION PROPOSÉE

5.2.1 Explications générales

Le problème à résoudre est donc la classification des pixels d'une image. Il existe plusieurs approches pour résoudre ce problème :

- Segmentation sémantique : les classes sont attribuées aux pixels de l'image en même temps. Les classes sont attribuées à la suite d'un problème d'optimisation à l'échelle de l'image
- Classification par pixel : une classe est attribuée à chaque pixel de l'image de manière indépendante

Le type de modèle utilisé pour la segmentation requiert beaucoup plus de paramètres et de temps de calcul. En particulier, le modèle ne profite que de très peu d'invariances (rotation, échelle, …) utilisées pour réduire le nombre de paramètres à estimer. En parallèle, la classification utilise des modèles plus simples, plus portables, et plus simples à estimer. La classification par pixel est choisie pour la suite des analyses.

Pour chaque pixel, une information sur son contexte est extraite de l'image puis utilisée pour faire la classification. L'information de contexte prend la forme d'un voisinage centré sur le pixel que l'on nomme patch. Cette image est utilisée comme entrée du système de classification. La classe prédite pour un pixel est alors définie à l'aide d'une prédiction calculée sur un patch centré sur le pixel et par conséquent, la prédiction d'une image est obtenue par la prédiction d'un patch glissant sur l'image originale (cf Figure 65). Contrairement à la segmentation, la classification profite de beaucoup plus d'invariance : en particulier, l'invariance par rotation pour les images satellite sera utilisée. Concrètement, la rotation d'un patch ne doit pas affecter sa classification. Notons que la taille du patch utilisé en entrée est un facteur extrêmement important puisque des études ont montré que l'augmentation de cette taille permet d'apprendre des descripteurs plus complexes et augmente les performances (Mnih 2013). Cependant, la taille du patch ne peut pas être trop grande pour conserver la précision et maintenir la quantité de calcul à une valeur raisonnable. La taille du patch a été dimensionnée afin d'être similaire à des problèmes d'apprentissage profond connus et résolus (Krizhevsky A., Sutskever I. et Hinton G. 2012).

Notons que la donnée bien segmentée est plus couteuse à obtenir que la donnée classifiée car elle nécessite des outils et des connaissances dédiés.

5.2.2 Formulation proposée

Nous nous proposons dans ce chapitre de résoudre ce problème grâce à la formulation du problème de classification de patches d'image suivant. Notons $\mathcal{I} = \left(I^{(1)}, \dots, I^{(N_\mathcal{I})}\right)$ la base de données d'images SPOT 6 album contenant $N_\mathcal{I}$ images et $\mathcal{M} = \left(M^{(1)}, \dots, M^{(N_\mathcal{I})}\right)$ la base de données de masques associés qui dans notre cas seront des masques de nuages et des masques de terrain. Les images $I^{(\cdot)}$sont des images rectangulaires constituées de $c = 4$ bandes spectrales. Le masque $M^{(n)}$ a la même taille que son image

$I^{(n)}$ et l'on note $M_i^{(n)}$ le label du pixel i de l'image $I^{(n)}$. Pour nos cas d'utilisation, $M_i^{(n)}$ peut prendre les valeurs parmi les labels possibles soit $\{0,1\}$ pour la détection de nuages et $\{0, \dots, 17\}$ pour la classification des sols.

Afin de faciliter la lecture, les indices n sont mis de coté et I et M désigne de manière générale une image et un masque de la base de données. L'approche proposée propose de prédire les patches de M à partir de l'étude de patches de I soit :

$$p(M, i, w_m) \sim p(I, i, w)$$

où $p(I, i, w)$ désigne le patch $w \times w$ de I centré sur le pixel i.

Cela revient alors à modéliser la relation entre le patch de taille $w_m \times w_m$ du masque M centré sur le pixel i à partir du patch de taille $w \times w$ de l'image associée I centré sur le même pixel i. De même, on utilisera $w_s = w_m$ c'est-à-dire qu'un même contexte i.e. patch de l'image servira à prédire le contenu du patch de même taille du masque (Mnih 2013).

Pour conclure sur la phase de modélisation, l'effet de la proximité de deux patchs sur leurs prédictions est négligé et permet de formuler le problème comme un problème classique de classification et d'apprentissage.

Le problème d'apprentissage est alors défini tout d'abord par sa base de données :

$$S = (\boldsymbol{X}_i, y_i)_{i=1,\dots,N} = \left(p\left(I^{(n)}, l, w_s\right), g\left(p\left(M^{(n)}, l, w_m\right)\right)\right)_{n=1,\dots,N_{\mathcal{I}}, l=1,\dots,N_{w_s}^{I^{(n)}}}$$

qui est alors constituée de tous les patches $p\left(I^{(n)}, l, w_s\right)$ de taille w_s de chaque image $I^{(n)}, n = 1, \dots, N_{\mathcal{I}}$ centrés sur tous les pixels $l = 1, \dots, N^{I^{(n)}}$ de l'image $I^{(n)}$ qui compte un total de $N^{I^{(n)}}$ pixels, et g est une fonction qui calcule des statistiques sur le patch du masque et renvoie un indicateur de classe suivant ces statistiques. Par exemple, g peut renvoyer la classe majoritaire dans le patch de masque. Les différentes formulations de g pour notre cas d'utilisation seront explicitées plus loin.

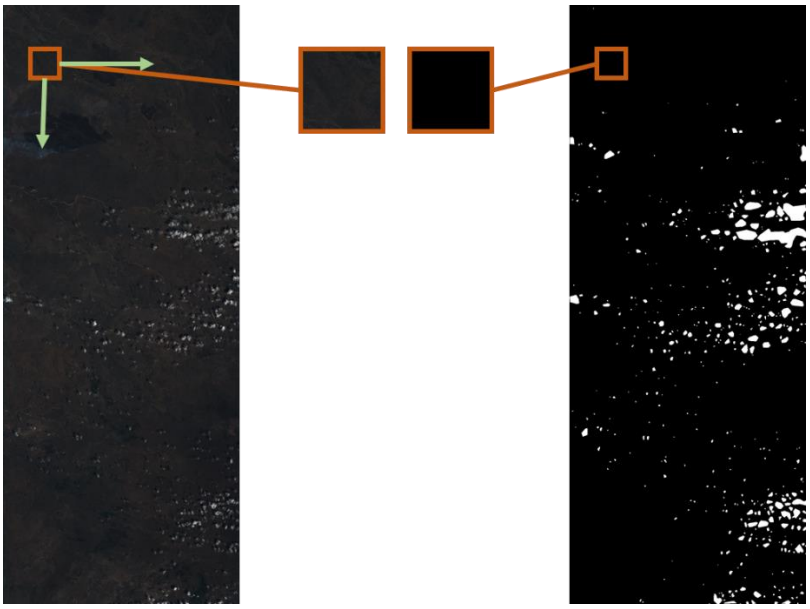

Figure 65: Schéma de fonctionnement de l'analyse par patch

5.2.3 Post traitements

La formulation actuelle sous la forme d'un problème de classification est connue pour créer des artefacts lors de la prédiction (Farabet C., et al. 2013). En effet, les prédictions sur de patches adjacents sont indépendantes et cette indépendance introduit du bruit de classification lors de la génération de la prédiction d'une image entière. Des traitements additionnels sont généralement appliqués sur le masque de prédiction généré pour améliorer les performances et produire des meilleures prédictions. Par exemple, les auteurs de (Farabet C., et al. 2013) proposent d'utiliser des superpixels (groupes de pixels proches et similaires) pour agréger les prédictions. Les prédictions finales sont alors attribuées par groupe de pixels proches. Des techniques plus avancées permettent de prendre de décisions globales au niveau de l'image en résolvant un problème d'optimisation se rapprochant d'une segmentation. Par exemple, les auteurs de (Farabet C., et al. 2013) définissent un graphe de similarité sur les super pixels pour attribuer les prédictions finales. Notons que les techniques de post traitement dépendent en grande partie des prédictions des pixels et ne sont pas considérées dans cette thèse (Farabet C., et al. 2013).

5.2.4 Discussions

L'utilisation d'une étape de classification de patches est courante pour l'analyse de l'imagerie spatiale (Audebert N., Le Saux B. et Lefevre S. 2016) (Tuia D., et al. 2011). Afin de traiter les images satellite qui sont habituellement grandes, les traitements sont habituellement appliqués sur des patchs d'images. La formulation actuelle du problème se rapproche des réseaux « fully convolutional » où la génération du masque est réalisée à partir de l'application d'un même réseau sur une fenêtre glissante sur l'image (Long J., Shelhamer E. et Darell T. 2015). L'évolution naturelle de la structure actuelle est d'utiliser des réseaux dédiés pour la segmentation (Audebert N., Le Saux B. et Lefevre S. 2016) (Ronneberg O., Fischer P. et Brox T. 2015) ou d'opter pour une approche par proposition de régions (Audebert N., Le Saux B. et Lefevre S. 2016) (Ren S., et al. 2015). La proposition de régions consiste à introduire une première phase de détection

de candidats potentiels pour la classification et le réseau final de classification est appliqué qu'aux candidats sélectionnés. Les deux approches requièrent une modification importante de la structure actuelle et en particulier, la croissance de la taille des réseaux utilisés qui s'avèrera par la suite complexe.

L'utilisation d'une phase de post-processing a déjà été utilisé conjointement à l'utilisation de réseau profond. En particulier, en imagerie spatiale où les pixels voisins sont statistiquement similaires, l'agrégation de performance par régions améliorent les performances des classifieurs. On peut cependant noter que l'extraction des régions peut être réalisée de différentes manières et les méthodes de segmentation utilisées pour détecter les régions peuvent varier (Audebert N., Le Saux B. et Lefevre S. 2016).

Enfin, une seule échelle de patch à savoir 32×32 comme les images de la base CIFAR est utilisée dans cette modélisation mais une étude multi échelle serait indiquée pour le problème de classification (Audebert N., Le Saux B. et Lefevre S. 2016) (Farabet C., et al. 2013). Une structure utilisant un ou plusieurs réseaux travaillant sur plusieurs échelles a déjà été utilisée pour réaliser de la labélisation sémantique d'images avec succès. Cependant, afin de simplifier dans un premier temps l'analyse et dans un deuxième temps la quantité de calcul, seule une échelle a été considérée dans cette analyse.

5.3 APPRENTISSAGE PROFOND

Comme il a été expliqué dans l'état de l'art et dans l'introduction, l'apprentissage est un candidat idéal pour résoudre ce type de problème de classification d'images. La partie suivante explique l'application de l'apprentissage profond à ce problème et en particulier, définit les éléments structurels choisis pour résoudre le problème posé.

Des réseaux de neurones profonds sont utilisés car ils permettent d'apprendre une structure hiérarchique de descripteurs adaptés optimisés pour le problème cible (Bengio Y., Courville A. et Vincent P. 2013). Une définition exacte des réseaux de neurones profonds a déjà été effectuée lors de l'état de l'art (cf 3.4.4). A titre de rappel, un problème d'apprentissage profond pour la classification est défini par :

- Des données : un ensemble d'exemples contenant des exemples et la classe associée
- Une fonction de coût : qui définit le problème d'optimisation à résoudre
- Une structure de réseau : qui définit la topologie du réseau de neurones utilisé dont les paramètres sont à estimer
- Un algorithme d'optimisation : qui définit la manière dont les paramètres sont estimés

5.3.1 Base de données

La base de données utilisée est la même base d'images que celle utilisée pour les expériences précédentes. Pour rappel, elle est constituée d'images album SPOT 6 à 60m de résolution et leurs masques de nuages associés. Comme il a été présenté précédemment, le problème est généralement simplifié de manière à classifier des patchs. Pour cela, des patchs sont extraits des images et une classe leur est attribuée pour chaque cas d'utilisation. Le mode d'attribution des labels suivant chaque cas d'utilisation sera expliqué prochainement.

Plus la taille du patch est grande, plus les descripteurs sont complexes et inversement, une taille trop petite empêchera l'apprentissage. En revanche, la taille du patch a des conséquences négatives : par exemple, la précision des labels est affaiblie puisque le label majoritaire change et ne signifie plus la même chose. De la même manière, un recouvrement est nécessaire pour lisser les prédictions. Dans le but de déterminer la taille de patch, une étude a d'abord été effectuée sur les tailles des images habituellement utilisées par les réseaux profonds (Krizhevsky A., Sutskever I. et Hinton G. 2012) (Simonyan K. et Zisserman A. 2015). Des tailles de l'ordre d'une centaine de pixels sont majoritairement utilisées (Simonyan K. et Zisserman A. 2015) mais de telles tailles ne seraient pas possibles pour nos cas d'utilisation car elles nécessiteraient beaucoup de calculs. Cependant, des réseaux convolutionnels ont également montré de très bonnes performances sur des images de taille 32×32 lors de l'étude de la base CIFAR-10 (Farabet C., et al. 2013) (Krizhevsky A., Sutskever I. et Hinton G. 2012) (Graham 2014). Dans le but de pouvoir bénéficier des structures ayant fonctionnées sur CIFAR, une taille de patch est fixée à 32 dans cette thèse.

5.3.1.1 Pré traitements

Pour des problèmes d'apprentissage automatique, les données sont normalisées de manière à ce que chaque valeur de descripteur varie dans le même intervalle. Cela permet d'éviter à l'apprentissage d'inclure le facteur d'échelle dans les poids à estimer. Classiquement, les valeurs par descripteurs sont normalisées de manière à avoir des données de moyenne nulle et de variance 1. Dans le cas d'images, les images en entrée sont souvent normalisées par image c'est-à-dire que l'image initiale est corrigée par sa moyenne et son écart type (Krizhevsky A., Sutskever I. et Hinton G. 2012) (Simonyan K. et Zisserman A. 2015). Cette méthode est également connue sous le nom de normalisation de contraste par image. Cette transformation est particulièrement intéressante pour des images car elle permet de supprimer les effets des conditions d'illumination tout en maintenant les relations entre bandes (cf Figure 66). D'autres prétraitements traitements existent (Krizhevsky A., Sutskever I. et Hinton G. 2012) mais dans un souci de simplification, seul ce prétraitement de données est utilisé dans toutes les simulations effectuées au cours de cette thèse.

Soit I une image et $\tilde{I}$ l'image normalisée, on peut écrire l'opération de normalisation sous la forme suivante :

$$\tilde{I} = \frac{I - \bar{I}}{\sigma(I)}$$

Où $\bar{I}$ est la moyenne de I et $\sigma(I)$ son écart type.

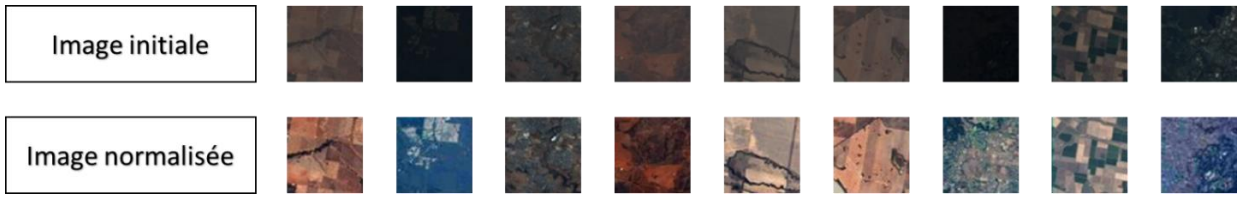

Figure 66: Prétraitement appliqué aux données.

5.3.1.2 Augmentation de données

La puissance de calcul actuelle permet de construire et estimer de très grands réseaux de neurones profonds jusqu'à des centaines de milliards paramètres. La quantité de données d'apprentissage est cruciale pour estimer de tels réseaux. Beaucoup d'auteurs (Howard 2014) (Krizhevsky A., Sutskever I. et Hinton G. 2012) (Wu R., et al. 2015) estiment que l'augmentation de données est fondamentalement importante pour améliorer les performances des réseaux. L'augmentation de données consiste à augmenter artificiellement la taille de la base de données d'apprentissage en ajoutant des nouveaux exemples crées à partir de déformations des exemples initiaux. L'objectif est que le réseau apprenne des descripteurs spécifiques aux classes d'objets considérées plutôt que des artefacts d'images comme des différences d'illuminations.

Par exemple, pour des images classiques, il est clair qu'un objet ne change pas si l'éclairage ambiant change ou si l'observateur est remplacé par un autre. En particulier, pour les images satellites, la présence d'un objet comme les nuages dans une image satellite est indépendant de la rotation de l'image. Le modèle final doit être moins sensible aux couleurs qui sont déterminées par les conditions d'illumination de la scène pour les images classiques et à l'orientation de l'image qui est causée par les conditions d'acquisition du satellite pour les images satellites.

L'augmentation de données consiste à appliquer certaines transformations sur l'ensemble initial d'apprentissage pour créer de nouveaux exemples artificiels. Les transformations appliquées ne changent pas la nature de la classe détectée et ainsi créent de nouveaux exemples. L'augmentation de données permet de créer artificiellement des invariances du réseau final et permet d'augmenter les performances en généralisation (Howard 2014) (Wu R., et al. 2015).

Pour des images classiques, parmi les opérations classiques utilisées pour l'augmentation des données, on peut trouver : la rotation, l'agrandissement et la translation (comme le montre la Figure 67). Des transformations plus avancées comme une modification du contraste ou de la luminosité de l'image peuvent être également ajoutées selon les besoins de l'utilisateur.

Les images satellites sont plus spécifiques que les images classiques et ne bénéficient pas d'autant d'invariances : la principale invariance d'une image satellite est l'invariance par rotation. Les différentes rotations de 90° sont appliquées et une symétrie verticale permet d'augmenter la base de données par un facteur 8 (cf Figure 68). Des rotations supplémentaires pourraient être rajoutées mais elles nécessiteraient de modifier la structure actuelle d'extraction de patchs. En effet, une rotation d'un angle non multiple de 90° nécessiterait une étape d'interpolation et l'utilisation de pixels à l'extérieur du patch considéré.

Figure 67: Exemple d'augmentation de données réalisées pour une image classique : flip horizontal, décalage, scaling, … Source

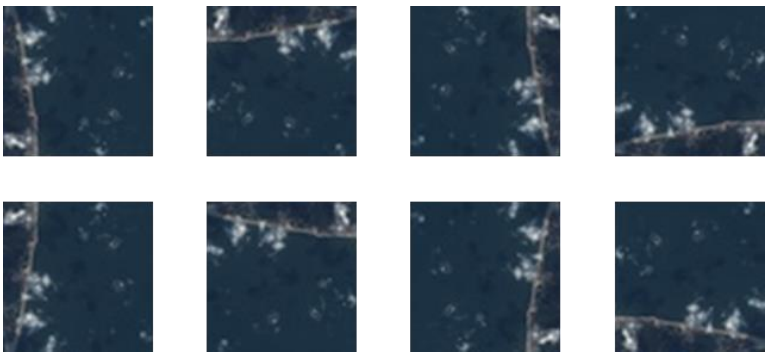

Figure 68: Augmentation de données appliquée à un patch d'apprentissage

5.3.1.3 Génération des labels

Les données utilisées à traiter ayant été constituées, il reste à déterminer la quantité à prédire. Pour rappel, l'objectif est de prédire une quantité sur le patch de masque cible à partir du patch de l'image associée. Comme il a été évoqué dans (Mnih 2013), il est plus performant de prédire des statistiques sur des patchs de masque que de prédire des labels de pixels à partir d'un même contexte. De plus, afin de simplifier les calculs, on veut ramener le problème à un problème de classification. Donc des labels sont créés à partir des statistiques du patch de masque. Par exemple, pour la classification de sols, le label doit permettre de répondre à la question : de quel type de sol est constitué ce patch ? Le label est alors attribué grâce l'empreinte du patch sur une carte de terrain (comme celle présentée dans la Figure 15), si une ambiguïté existe, i.e., la carte fournit plusieurs types de terrains présents sur le patch, la catégorie du terrain majoritaire est utilisée pour le label. Le cas de la détection de nuages pourrait être résolu de la même manière mais la précision du système peut être améliorée en rajoutant des classes. Trois 3 catégories sont créées à partir de la couverture nuageuse du patch (pourcentage de pixels appartenant à des nuages, cf Figure 69). Si la couverture nuageuse est de 0%, la première catégorie correspond aux patchs dépourvus de nuages. Si la couverture nuageuse est strictement positive, la deuxième catégorie contient des patchs avec présence d'un nuage. Et la dernière catégorie correspond aux patchs recouverts à plus de 50% de nuages.

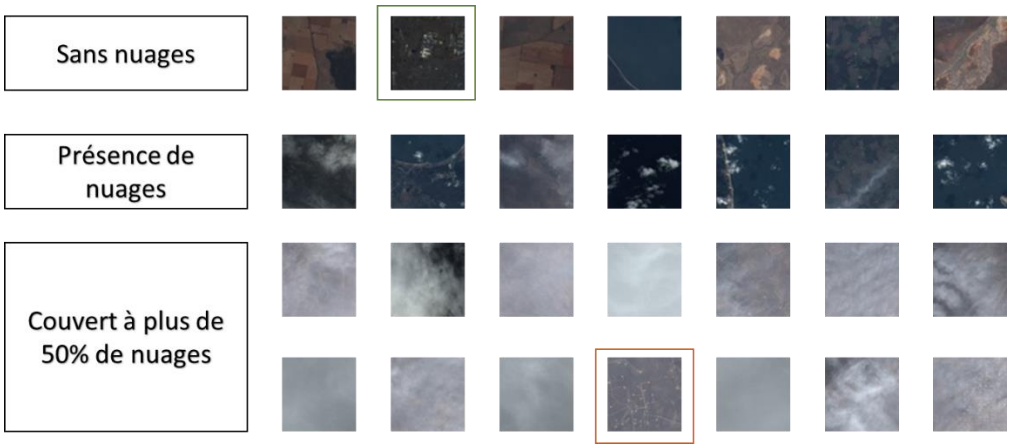

Figure 69: Exemples de patches par label. Vert : artefact ressemblant à des nuages. Rouge : Brume.

5.3.2 Fonction de coût

La formulation et les approximations présentées précédemment ont permis de simplifier le problème à un problème de classification. Les données sont $S = (\boldsymbol{X}_i, y_i)_{i=1,\dots,N}$ où $\boldsymbol{X}_i$ est un patch d'apprentissage et y_i sa classe associée. Notons K le nombre de classes, f le réseau profond tel que $\boldsymbol{P}_i = f(\boldsymbol{X}_i)$ soit le vecteur de probabilité issu du réseau où $\boldsymbol{P}_i^k$ est la probabilité associée à la classe k, et $\boldsymbol{Y}_i$ le vecteur d'indicatrices contenant un 1 à la position associée à la classe y_i et des zéros ailleurs $\boldsymbol{Y}^{y_i} = 1, \boldsymbol{Y}^k = 0 \; si \; k \neq y_i$. La fonction de coût associée à ce problème de classification est l'entropie croisée et s'écrit :

$$L(\boldsymbol{P}, \boldsymbol{Y}) = -\sum_{k=1}^{K} \boldsymbol{Y}^k \log \boldsymbol{P}^k$$

L'entropie croisée est souvent utilisée pour l'apprentissage de réseau profond car elle est parfaitement adaptée au problème de classification (Friedman J., Hastie T. et Tibishirani R. 2001).

5.3.3 Algorithme d'optimisation

On rappelle que la partie résumant l'apprentissage supervisé a permis de résumer la classification à trouver le meilleur modèle f_θ paramétré par l'ensemble de poids θ minimisant le risque empirique $\epsilon(S, \theta)$ qui est la moyenne de la fonction de coût sur la base d'exemples S (cf 3.4.1):

$$\epsilon(S, \theta) = \sum_{i=1}^{N} \frac{1}{N} L(f_\theta(\boldsymbol{X}_i), \boldsymbol{Y}_i)$$

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \, \epsilon(S, \theta)$$

En apprentissage profond, comme il a été expliqué précédemment, les poids sont estimés grâce à une descente de gradient stochastique de la manière suivante, où η est le taux d'apprentissage :

$$\theta_{t+1} \leftarrow \theta_t + \eta \frac{\partial L(f_\theta(\boldsymbol{X}_l), \boldsymbol{Y}_l)}{\partial \theta}\Big]_{\theta=\theta_t}$$

η est appelé le taux d'apprentissage et est un paramètre extrêmement sensible pour la convergence de la méthode (Bottou L. 2012). Cette procédure est exécutée pendant T itérations c'est-à-dire que les paramètres sont mis à jour T fois en utilisant T exemples. On utilise souvent le nombre d'epochs $\frac{T}{N}$ pour caractériser la durée d'apprentissage. Une epoch correspond à N mises à jour soit l'équivalent du jeu de données entier. Cependant, bien que cette méthode ait des garanties intéressantes de convergence, elle est très sensible au taux d'apprentissage (en particulier sur sa valeur initiale (Bottou L. 2012)) et l'estimation du gradient souffre d'une grande variance (voir figure suivante).

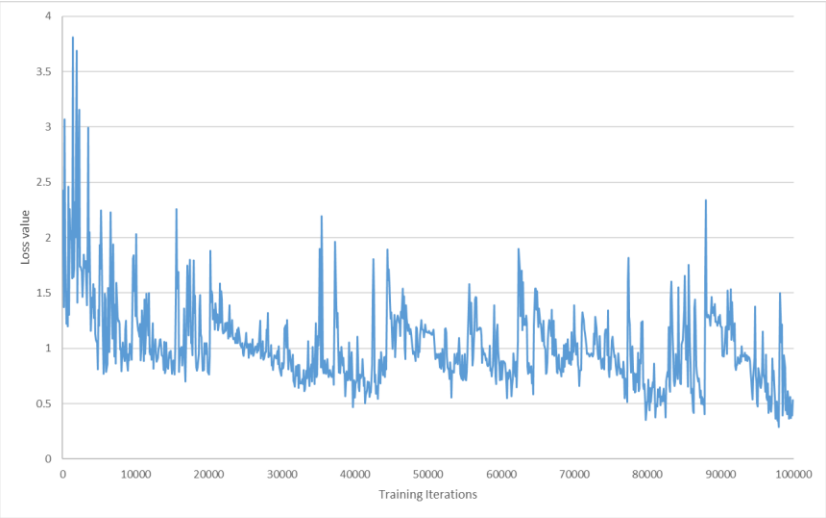

Figure 70: Valeur de la fonction de coût lors d'une descente de gradient stochastique pour un réseau profond.

Des techniques ont donc été développées pour atténuer ces perturbations : une d'entre elles consiste à introduire la notion de moment comme le propose la méthode Adam. Un moment consiste à ajouter l'information des derniers mouvements (les dernières mises à jour) dans la mise à jour courante : ainsi, on garde la tendance des précédentes mises à jour et on « lisse » la descente. De manière pratique, si plusieurs mises à jour suivent la même direction, un moment est créé dans cette direction c'est-à-dire que le mouvement s'accélère dans cette même direction au fur et à mesure des mises à jour. A l'inverse, si une mise à jour change de direction par rapport aux précédentes, le déplacement sera ralenti. De plus, la seconde force de la méthode Adam est de calculer des taux d'apprentissage adaptatifs pour chaque paramètre.

En pratique, l'introduction de moment dans la méthode signifie qu'une fraction de la précédente mise à jour est ajoutée à la mise à jour courante. Pour cela, la méthode Adam calcule deux quantités : m_t la moyenne courante de la dérivée de la fonction de coût g_t et v_t la moyenne courante de g_t^2. Les deux moyennes sont dites courantes car une proportion de leur valeur précédente est conservée. Ces quantités sont définies comme suit :

$$g_t = \frac{\partial L(f_\theta(\boldsymbol{x}_i), y_i)}{\partial \theta}\Big]_{\theta_t}$$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

m_t et v_t sont des estimations du premier moment (la moyenne) et du deuxième moment (la variance) des gradients respectivement, d'où le nom de la méthode. Ce qui engendre la méthode de mise à jour suivante :

$$\theta_{t+1} \leftarrow \theta_t - \frac{\eta}{\sqrt{v_t} + \epsilon} m_t$$

En particulier, le terme de déplacement est $\eta \frac{m_t}{\sqrt{v_t}}$ [6] dans le domaine des paramètres. Schématiquement, $\frac{m_t}{\sqrt{v_t}}$ représente la direction de déplacement et η la norme du déplacement. La direction est calculée à partir de celle m_t qui est la moyenne courante de g_t et le déplacement est alors maximisé lorsque les déplacements successifs sont dans le même sens et ne s'annulent pas.

L'auteur de l'article initial (Kingma D. et Ba J. 2015) propose $\beta_1 = 0.999 = \beta_2$ et $\epsilon = 10^{-8}$ et l'article montre que sur un grand nombre d'expériences, la méthodologie fonctionne et même surpasse les autres méthodes adaptatives.

De la même manière, la variance de l'estimation du gradient g_t peut être diminuée en calculant un autre estimateur basé sur la moyenne de g_t sur plusieurs exemples, constituant un « batch » de données. Ainsi, l'optimisation ne se fait plus en itérant sur tous les exemples de la base de données mais sur des ensembles de taille fixe que sont les batchs de données. Notons n_{batch} le nombre d'exemples du batch et $\mathcal{B} = \left(\boldsymbol{b}_j, c_j\right), j = 1, \dots, n_{batch}$ les éléments du batch, on définit le gradient moyen par batch par :

$$g_t = \frac{1}{n_{batch}} \sum_{j=1}^{n_{batch}} \frac{\partial L\left(f_\theta\left(\boldsymbol{b}_j\right), c_j\right)}{\partial \theta}\Big]_{\theta_t}$$

La taille de cet ensemble $\mathcal{B}$ a une grande influence sur les performances finales. L'utilisation d'un batch (Bottou L. 2012) devait dans un premier temps aider à lisser la courbe d'apprentissage de la Figure 70. Mais une analyse plus fine de l'utilité de cet ensemble a permis de montrer qu'un compromis entre la taille de cet ensemble et les performances finales doit être trouvé (Goyal P., et al. 2017). L'estimation fine du gradient sur un exemple permet en réalité d'éviter de tomber dans des minimums locaux (ou d'en sortir), au contraire de l'approche consistant à utiliser des batchs de taille importante. De récents travaux (Goyal P., et al. 2017) ont montré grâce à des expériences répétées que l'on peut observer un décrochage (observé sur la Figure 71) de la précision avec l'utilisation d'une taille de batch trop élevée sur leurs expériences.

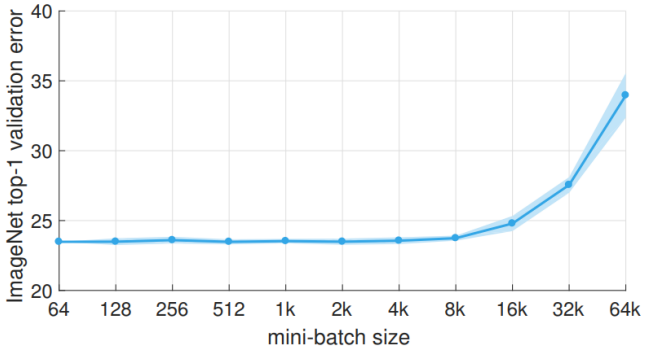

Figure 71: Analyse de la performance finale en fonction de la taille de batch utilisée lors de l'optimisation. (Goyal P., et al. 2017)

Pour conclure, un dernier paramètre reste à déterminer. Le chapitre décrivant les réseaux profonds (cf 3.4.4.6) a permis d'introduire la notion de dropout (Srivastava N., et al. 2014) qui consiste à éliminer certaines connexions du réseau lors de l'apprentissage. De fait, la structure du réseau change entre deux itérations et est supposée éviter le sur apprentissage. Le dropout est principalement appliqué aux couches

[6] $\epsilon = 0$ facilite l'interprétation

connectées pour notre cas d'utilisation. Cependant, le dropout nécessite un paramètre à savoir le nombre d'unités à éliminer à chaque itération qui est souvent exprimé en tant que taux d'unités à conserver. Par exemple, un taux de 10% éliminera 90% des connections. Ce paramètre doit également être choisi pour maximiser les performances lors de l'apprentissage.

5.3.4 Structure des réseaux

Le dernier paramétrage à définir est la topologie du réseau à utiliser. En particulier, la topologie d'un réseau inclut sa structure : le nombre de couches de convolutions, la taille des noyaux de convolution, le nombre de couches connectées et leur taille.

5.3.4.1 Réseaux à une ou deux couches

Le premier type de réseau que nous avons utilisé constitue notre méthode de référence qui a montré des résultats intéressants pour la détection de nuages (Chandran A. et Christy J. 2015). Des descripteurs numériques (ratios (Hollstein A., et al. 2016), Gabor, et DCTs (Noureldin L., et al. 2012)) sont extraits des images comme il a été fait pour le chapitre précédent et sont utilisés comme entrées pour les réseaux de neurones. Les réseaux de neurones utilisés sont des réseaux qui prennent en entrée un vecteur d'attributs numériques et renvoient un vecteur de probabilités dont la dimension est la taille du nombre de classes souhaitées. Ces réseaux sont classiquement constitués de couches connectées et possèdent une ou deux couches cachées pour limiter leur complexité et le surapprentissage.

5.3.4.2 Réseaux profonds

Comme il a été expliqué, l'idée principale de ce chapitre est de définir les structures des réseaux convolutionnels utilisés dans cette thèse. L'apprentissage profond a pour principal avantage de faciliter le contrôle du nombre de poids à estimer du réseau. En réalité, le nombre de poids à estimer du réseau doit être adapté à la complexité du problème de classification à traiter. Un nombre de poids trop élevé rendra l'estimation complexe et aboutira à de faibles performances et un nombre de poids faible ne permettra pas de modéliser la complexité des relations entre l'entrée et la prédiction.

De plus, dans le cas présent, la taille de l'entrée va également limiter la complexité des réseaux à utiliser car la taille du vecteur d'entrée du réseau de 32×32 limite les relations et descripteurs que le modèle peut apprendre. La base de données publique la plus similaire est la base de données CIFAR-10 et 100 qui sont composée de 60000 images couleurs de taille 32×32 (Krizhevsky A., Sutskever I. et Hinton G. 2012). Cette base de données est toujours actuellement beaucoup étudiée et un grand nombre de structures différentes ont été testées pour cette base de données (Graham 2014) (He K., et al. 2016). Les structures utilisées pour cette base sont les plus à mêmes de fonctionner sur nos cas d'utilisation. Par exemple, une grande partie des réseaux existants sont conçus pour résoudre le problème de classification Imagenet (Deng J., et al. 2009) (Krizhevsky A., Sutskever I. et Hinton G. 2012) qui consiste à classifier des images de 224×224 qui sont beaucoup plus grandes que des imagettes 32×32. Ces réseaux sont plus profonds et possèdent plus de poids à estimer. Ainsi, ils sont moins adaptés à nos cas d'utilisation.

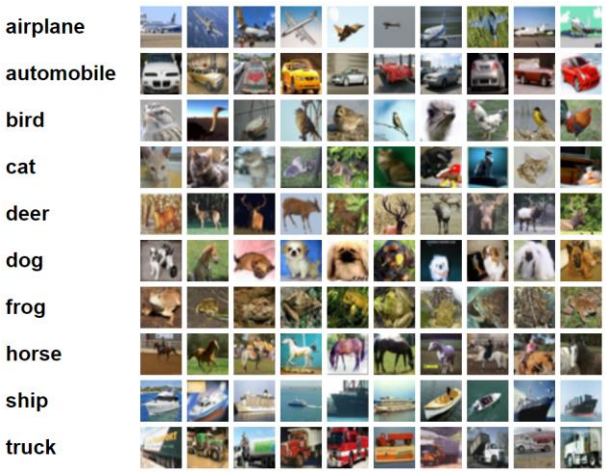

Figure 72: Exemples d'images et de classes contenues dans la base CIFAR. Source

La taille des images est également liée à la complexité des descripteurs que peut apprendre un réseau convolutionnel. En effet, intuitivement, des images basse-résolution (de petites tailles) ne permettent pas la détection de petits éléments tels que des yeux, etc.

Pour rappel, un CNN est constitué d'une succession de couches de convolution et de couches de sous échantillonnage. Une couche de sous échantillonnage diminue la taille de l'image d'entrée par un facteur minimum de 2. Avec un vecteur d'entrée de taille 32×32, le nombre maximum de couche de « pooling » est de 5. Cependant, les couches de convolution peuvent également introduire des effets de bords lorsqu'elles sont appliquées à de petites images comparativement à leur taille de noyau de convolution[7]. Il n'est pas intéressant d'appliquer une convolution 7×7 comme celle de VGG 16 (Simonyan K. et Zisserman A. 2015) (cf Figure 74) à une image 8×8 et il serait plus pertinent d'appliquer une couche connectée en substitution. Le nombre de sous échantillonnage est fixé à 2 pour obtenir une image 8×8 avant d'utiliser des couches connectées. La structure obtenue est similaire à des réseaux performants sur CIFAR-10/100 qui sont des ConvNets de petite taille (k=2, l=2, cf Figure 73 (Graham 2014)) :

Conv, Pool, Conv, Pool, FC, FC

L'optimisation de la structure est un domaine actif de recherche. Des bonnes pratiques et des blocs de calcul performants ont été proposés (Iandola F., et al. 2016) (Szegedy, et al. 2016) (Szegedy C., et al. 2015) mais l'optimisation structurelle c'est à dire de l'agencement des couches de convolutions et de pooling reste un problème non résolu à l'heure actuelle et est exclu des analyses de cette thèse. Notons que ce problème est souvent traité par une recherche aléatoire ou systématique des structures.

[7] Une interpolation est réalisée pour calculer les convolutions des pixels placés sur les bords. Par exemple, une image 8×8 est transformée en 16×16 pour une convolution 7×7.

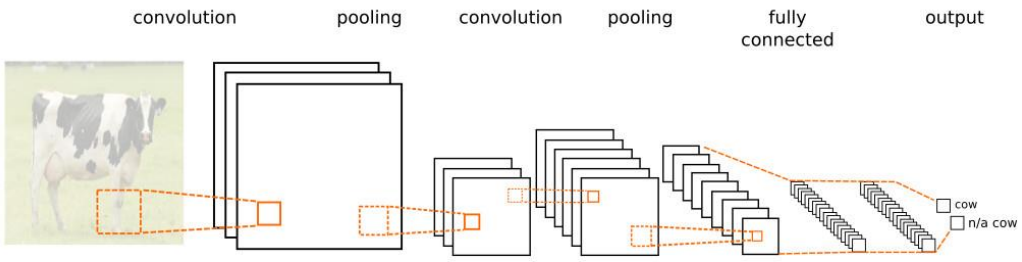

Figure 73: Structure ConvNet utilisée pour la base CIFAR 10 et l'analyse d'images 32x32. Source

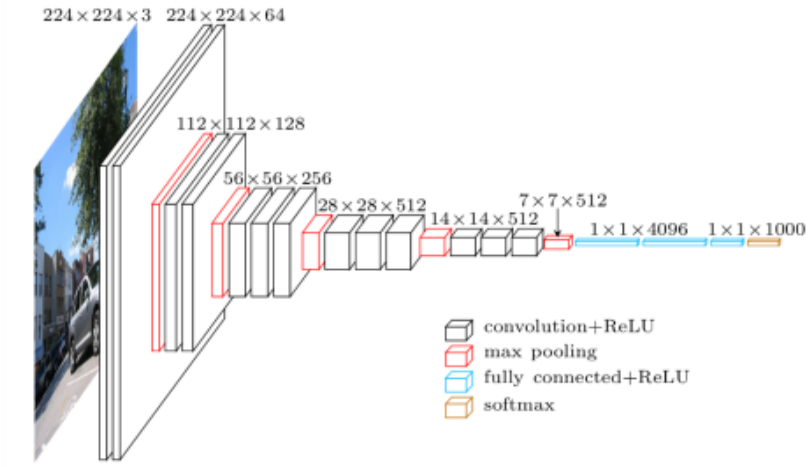

Figure 74: Structure du réseau VGG 16. (Simonyan K. et Zisserman A. 2015)

INPUT: [32x32x4] CONV-64: [32x32x64] weights: $(5 \times 5 \times 4) \times 64 = 6400$ POOL2: [16x16x64] CONV-64: [16x16x64] weights: $(5 \times 5 \times 64) \times 64 = 102400$ POOL2: [8x8x64] FC: [1x1x395] $8 \times 8 \times 64 \times 395 = 1617920$ FC: [1x1x192] $395 \times 192 = 75840$ FC: [1x1x10] $192 \times 10 = 1920$ TOTAL params: 1,8M parameters
Tableau 1: Nombre de paramètres d'un ConvNet (CifarNet)

Le Tableau 1 explique le calcul du nombre de poids à estimer dans un ConvNet et en particulier sur un réseau appliqué à CIFAR-10. Il est très important de remarquer qu'une grande partie des poids est concentrée sur les couches connectées. Ce constat explique le nombre conséquent de poids d'un réseau comme VGG 16 qui possède une couche cachée avec 4096 unités cachées et 139M de paramètres à estimer (Simonyan K. et Zisserman A. 2015). Comme il a été expliqué, le nombre de poids à estimer doit être adapté au problème et est modifiable par le biais du nombre de noyaux de convolutions, de la taille de ces noyaux et du nombre d'unités des couches cachées.

5.3.4.3 Hyperparamètres

Afin de pouvoir tester différentes structures, la paramétrisation suivante de la structure présentée précédemment est utilisée afin de tester différentes structures :

- La taille de noyaux de convolution utilisés
- Le nombre de convolutions à effectuer par couches de convolution.
- Le nombre d'unités cachées dans les couches connectées

Tous ces hyperparamètres ont une influence importante sur les performances et sur la répartition des poids du réseau : voir paragraphe précédent. La chaîne globale de traitement est représentée sur la Figure 75 et réunit toutes les étapes décrites précédemment.

5.3.5 Discussions

Les choix conceptuels précédents ont été réalisés pour simplifier les recherches et les premiers tests. Le monde de l'apprentissage profond est un monde basé sur des démonstrations empiriques et dont la constante de temps est extrêmement rapide. La conséquence est l'ignorance d'une méthodologie globale de résolution des problèmes : en effet, les expériences sont réalisées sur des cas d'utilisation spécifiques et il n'existe pas de preuve que les performances soient transférables à d'autres cas d'utilisation. La constante de temps de l'arrivée de nouvelles techniques dans le monde du « deep learning » est de l'ordre de quelques mois (2 ou 3) : il est donc difficile d'être à la pointe de la technologie en permanence. La réalisation des expériences pour certains choix implique de figer les technologies utilisées pour un temps et ces techniques seront alors « dépassées » lors de la fin des expériences.

Mis à part ce constat, certains choix faits pour nos expériences sont discutables. Premièrement, le choix de l'optimiseur Adam peut être remis en question : à l'époque des premières expériences, il était le plus utilisé et semblait être le plus performant en moyenne sur plusieurs cas d'utilisation. Cependant, ce choix était empirique et il est évident que l'algorithme d'optimisation doit être choisi de manière optimale pour chaque cas d'utilisation. Comme il a été présenté, un compromis entre simplicité et complexité d'utilisation a été trouvé en utilisant Adam. En ce qui concerne l'augmentation de données, elle joue un rôle fondamental en apprentissage profond (Wu R., et al. 2015). En effet, contrairement aux descripteurs classiques connus comme les ratios ou les SIFT, les comportements des descripteurs appris par apprentissage profond ne sont pas connus. En particulier, on ne connaît pas leurs invariances : rotation, translation, … Au contraire, les classifieurs « conçus à la main » ont été construits pour être robuste à certaines transformations. Par exemple, les ratios ont été conçus pour être robustes aux conditions d'illumination. En apprentissage profond, ces invariances sont construites à partir de l'analyse d'une quantité suffisante de données ou par l'ajout de transformation dédiées au travers de l'augmentation de données. L'augmentation de données utilisée ici est satisfaisante du point de vue de la physique de l'image satellite mais pourrait être améliorée en ajoutant des opérations supplémentaires (Howard 2014) (Wu R., et al. 2015). Cependant, à la différence du domaine du traitement d'images web, les opérations d'augmentation de données pour les images satellite ne sont pas clairement définies et nécessiteraient un travail de recherche pour leur définition. Dans le cas présent, la quantité de données à traiter est supposée intégrer toutes les transformations possibles et ainsi, éviter le besoin de l'utilisation de l'augmentation de données et l'addition de calculs supplémentaires.

Enfin, les choix de structure sont les plus à même d'être contestés car des travaux ont déjà montré que l'on pouvait construire des structures plus performantes que celles utilisées ici. Des blocs tels que Inception, ou l'utilisation de blocks résiduels ont montré de meilleures performances que l'enchainement classique de couches de convolutions et de pooling et amélioreront les résultats de détection de nuages. Cependant, aucune structure basique n'a été testée sur le cas de la détection de nuages qui est un cas assez simple comparé à l'identification de 1000 classes dans des images quelconques (Krizhevsky A., Sutskever I. et Hinton G. 2012). D'un premier abord, les problèmes de classification étudiés ici ne semblaient pas assez complexes pour nécessiter de telles structures qui ont pour vocation à être testées dans une extension de ces travaux. De plus, la plupart des structures existantes nécessite une adaptation pour être appliquées à des images 32×32. Déjà expliqué précédemment, l'optimisation de la structure des réseaux n'est pas considérée dans cette thèse afin de simplifier les analyses.

5.4 EXPERIENCES

5.4.1 Description

Le but des expériences est de tester les propriétés des réseaux convolutionnels pour la détection de nuages et la classification de terrain. Tout d'abord, les premiers réseaux simples servent à simuler l'approche historique : des descripteurs tels que des ratios, les coefficients de Gabor et DCTs sont extraits de l'image pour former le vecteur en entrée de ces réseaux. 1000 images ont été utilisées pour l'apprentissage, ce qui correspond à 100 millions de pixels à traiter. Notons que 1 pixel sur 8 est conservé par image pour éviter la redondance des pixels car localement, les pixels ont des valeurs similaires. Sur ces mêmes images, des patchs de taille 32×32 sont extraits autour de ces mêmes pixels pour l'apprentissage.

Les performances des réseaux de neurones classiques et des réseaux convolutionnels sont évaluées sur une base de 250 images tirées aléatoirement de la base de données totale et qui n'appartiennent pas à l'ensemble d'apprentissage. Les performances sont mesurées grâce à la précision par pixel. La deuxième phase de l'expérience consiste à étudier la sensibilité de l'apprentissage aux différents hyperparamètres. Certains hyperparamètres sont sensibles pour l'apprentissage et l'optimisation : en particulier, le choix de la taille du batch et du taux d'apprentissage est important pour les performances finales. En particulier pour ces deux paramètres, des apprentissages sont lancés avec des configurations spécifiées pour la comparaison. Les configurations sont obtenues en testant des valeurs prédéfinies pour chaque hyperparamètre. Finalement, les derniers tests concernent la structure des réseaux à utiliser. La paramétrisation de la structure par le nombre de filtres de convolutions, leur taille, … permet de lancer des apprentissages pour différentes combinaisons de paramètres de structures et ainsi comparer leur performance. Beaucoup de structures différentes doivent être analysées : ce qui explique pourquoi les apprentissages ont été lancés en parallèle dans le cloud sur GPU.

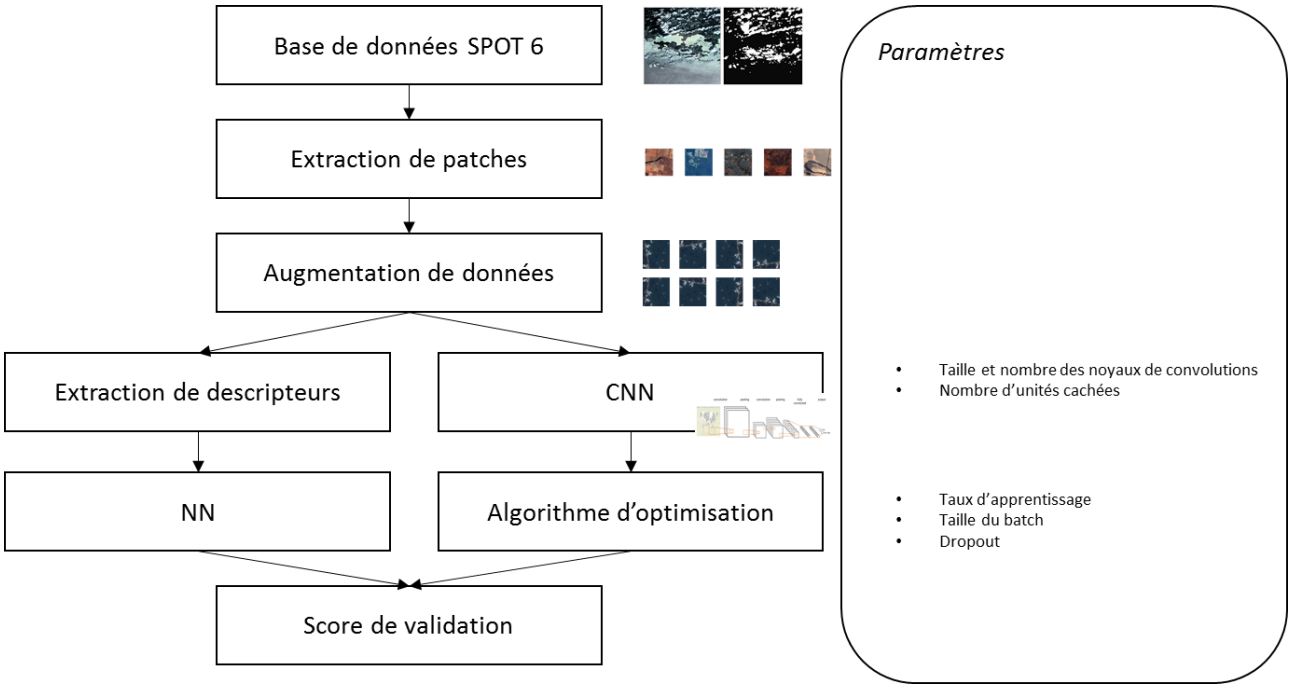

Figure 75 : Schéma récapitulatif de la méthodologie et des paramètres à optimiser.

5.4.2 Résultats

5.4.2.1 Analyse quantitative

5.4.2.1.1 Résultats quantitatifs

Les premiers résultats présentés dans la Figure 76 sont les scores de prédictions des différents réseaux suivant le type de descripteurs utilisés (Le Goff M., Tourneret J-Y et Wendt W., et al. 2017). Les performances du CNN sont les meilleures comparativement aux performances des autres réseaux et leurs descripteurs respectifs. Etant donné le nombre de paramètres à estimer élevés du CNN, il était attendu qu'il ait la meilleure performance. En revanche, on peut également remarquer que des descripteurs

connus et fixes comme les DCTs atteignent des performances similaires à celles des CNNs. Cependant, on peut également observer que tous les descripteurs n'ont pas le même niveau de performance : les coefficients de Gabor et les valeurs brutes des pixels offrent les performances les plus faibles. Les descripteurs servent à mieux encoder l'information que les valeurs brutes des pixels ainsi qu'à supprimer certains effets comme ceux produits par différentes conditions d'illumination : il est donc naturel que le calcul et l'utilisation de descripteurs tels que les ratios ou les coefficients DCT permettent d'augmenter les performances comparativement aux valeurs brutes des pixels. En revanche, on peut cependant noter que les coefficients de Gabor sont sensibles aux orientations et à la rotation ce qui peut expliquer leur faible performance. La structure par défaut du CNN utilisée est celle présentée dans la section précédente. Des changements pourraient améliorés les performances et sont étudiés par des expériences dédiées.

Input Type	Network	Accuracy	Recall	Precision
Gabor	ANN	77%	43%	66%
Raw Pixels	ANN	83%	68%	77%
Features	ANN	81%	68%	80%
Superpixels	ANN	83%	69%	80%
DCT	ANN	85%	75%	80%
Patches	CNN	86%	75%	81%

Figure 76: Comparaison des performances par descripteurs et structure utilisée.
(Le Goff M., Tourneret J-Y et Wendt W., et al. 2017)

La Figure 77 permet de visualiser les performances obtenues par les différentes méthodes de détection de nuages. Elle permet en particulier de visualiser les différences morphologiques des détecteurs. On retrouve en particulier un aspect pixelisé lors de l'utilisation de classifieur pixellaire (ratios et valeurs brutes) : ce phénomène est principalement dû à des erreurs de prédiction localisées et isolés. L'utilisation de patch ou de superpixels permet de lisser les prédictions mais l'utilisation des patchs et du CNN fournit de meilleures performances de détection.

Figure 77: Résultats de classification sur une image pour les différents descripteurs. En haut, de gauche à droite, l'image initiale, le masque de nuages, la prédiction avec les valeurs brutes. En bas, de gauche à droite, la prédiction avec les ratios, avec le CNN, avec les superpixels. (Le Goff M., Tourneret J-Y et Wendt W., et al. 2017)

Les performances du CNN sont meilleures que les descripteurs classiques mais celles-ci peuvent encore être augmentées en ajustant ses hyperparamètres. Tout d'abord, l'estimation des poids est issue d'un problème d'optimisation qui est résolu par descente de gradient stochastique avec la méthode d'Adam. Le premier paramètre à être étudié est le taux de dropout qui est pour rappel le taux d'unités aléatoirement éteintes durant l'apprentissage. Le taux de dropout a été principalement conçu pour éviter le sur-apprentissage et améliorer la généralisation des réseaux. En effet, la Figure 79 permet d'observer que l'erreur de généralisation est meilleure avec un taux de dropout important ici 10% c'est-à-dire que 90% des neurones des couches connectées sont éteints aléatoirement durant l'apprentissage. De même, on peut observer qu'un taux de dropout de 50% n'a que peu d'effet sur la courbe d'apprentissage de la Figure 78 et diminue les fluctuations de l'erreur de généralisation comparativement aux performances sans dropout. Cependant, on peut tout de même observer qu'un grand taux de dropout a un effet important sur les courbes d'apprentissage de la Figure 78. En particulier, plus un nombre important d'unités sont désactivées (90% pour un taux de dropout de 10%), plus la classification est complexe et par conséquence, la fonction de coût est plus élevée comme on peut le voir sur la Figure 78. En pratique, même si la fonction de coût est plus élevée, la performance plus importante et stable du réseau avec 10% de dropout (visible sur la Figure 79) est préférée aux autres et le taux de dropout est fixé à 10% pour la suite.

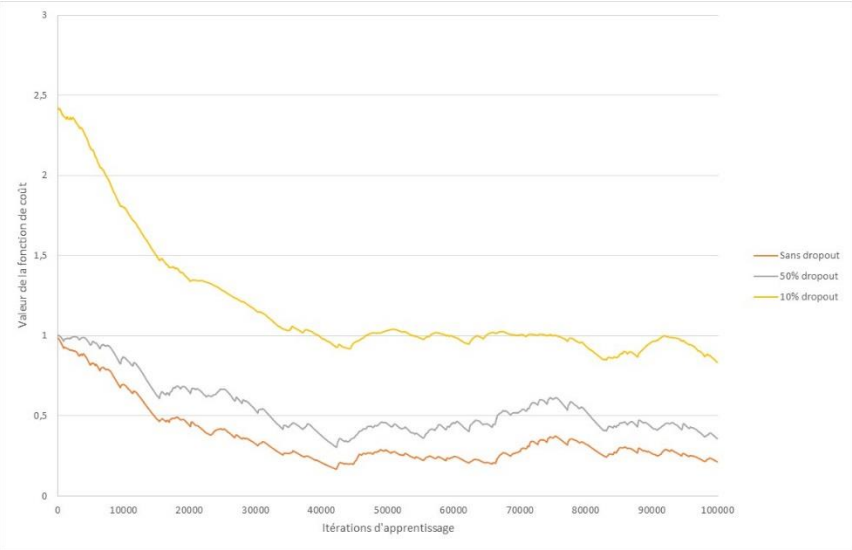

Figure 78: Graphique de la fonction de coût au cours de l'apprentissage pour différentes valeurs du taux de dropout.

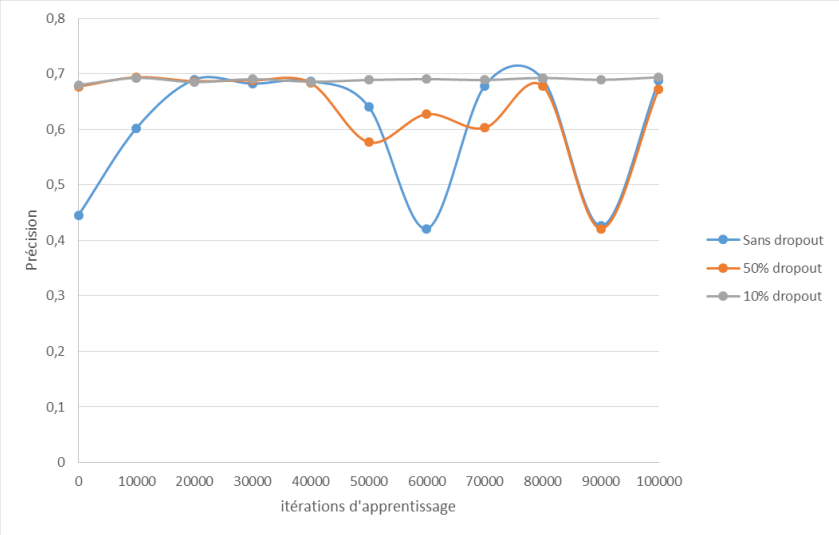

Figure 79: Score de validation au cours de l'apprentissage pour différentes valeurs de dropout.

Le second paramètre qui a une grande importance dans l'optimisation et le résultat de l'estimation est le taux d'apprentissage. Différentes valeurs ont été testées pour cette grandeur et la Figure 80 permet de visualiser la fonction de perte évaluée sur un batch de données au cours des 100000 premières itérations d'apprentissage pour le problème de la détection de nuages. La Figure 80 permet de mettre en évidence que les valeurs du taux d'apprentissage ont une grande influence sur la convergence de la méthode. Par exemple, la Figure 81 et la Figure 80 permettent de mettre en évidence que la valeur du taux d'apprentissage influe sur la vitesse de convergence. En effet, les courbes de la Figure 81 correspondant aux taux d'apprentissage faibles 10^{-7} et 10^{-6} convergent lentement mais de manière stable au contraire des courbes correspondant aux taux 10^{-4} et 10^{-5} qui convergent plus rapidement mais de manière

instable. Cependant, on peut remarquer qu'il n'y a plus de convergence si le taux d'apprentissage est trop élevé (10^{-3} et 10^{-2}) sur la Figure 81. Ce constat est logique car le taux d'apprentissage a pour vocation de ralentir la convergence afin d'éviter les minimas locaux. On peut tout de même noter que les valeurs élevées de taux d'apprentissage (supérieures à 10^{-5}) de la Figure 81 ont une variance importante qui peut limiter les performances de généralisation.

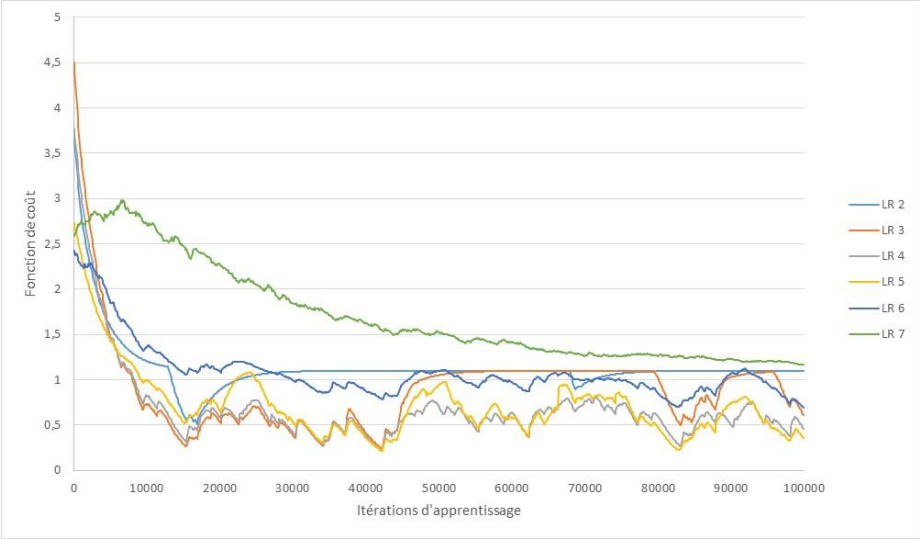

Figure 80: Graphique de la fonction de perte au cours du temps pour différentes valeurs du taux d'apprentissage. Légende : LR n signifie $\eta = 10^{-n}$

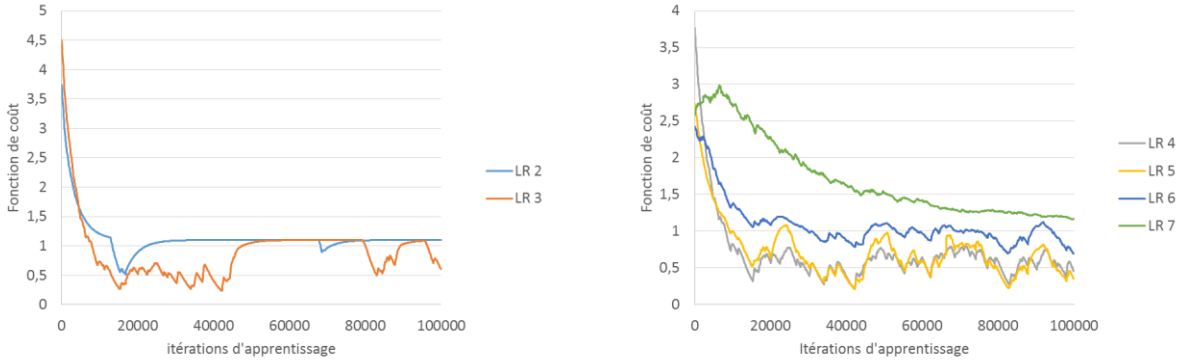

Figure 81: Graphique de la fonction de coût au cours de l'apprentissage pour différentes valeurs de taux d'apprentissage. Gauche : Taux élevés. Droite : taux plus faibles. Légende : LR n signifie $\eta = 10^{-n}$

En complément de la figure précédente, la Figure 82 montre le score de classification sur un échantillon de validation (non inclus dans la base d'apprentissage) des classifieurs obtenus avec les différents taux d'apprentissage. Dans l'ordre croissant, les performances en généralisation sont maximales pour 10^{-6} et 10^{-5}. Notons l'instabilité des performances de la courbe du taux d'apprentissage de 10^{-4} qui semble indiquer que le taux d'apprentissage est trop élevé. On peut également clairement observer une limite de performance de généralisation à 70% sur cette figure. Les analyses précédentes ont donc permis de choisir un taux d'apprentissage de 10^{-6} pour les futures expériences.

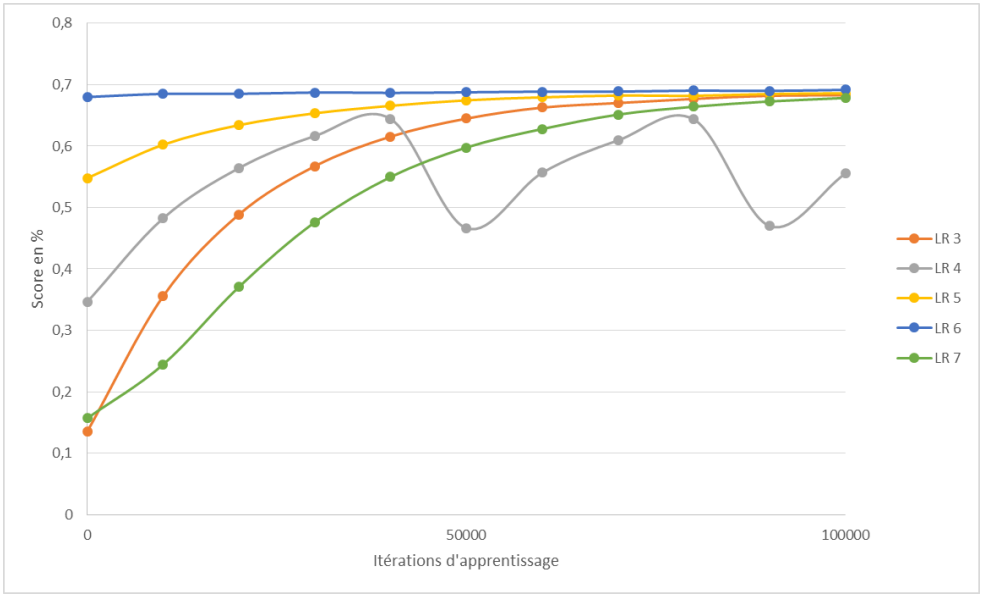

Figure 82: Graphique du score de validation au cours de l'apprentissage pour différentes valeurs du taux d'apprentissage. Légende : LR n signifie $\eta = 10^{-n}$

De la même manière que pour le taux d'apprentissage, l'influence de la taille du batch sur les performances d'optimisation doit être établi. Il a déjà été évoqué que des récentes études ont montré que la taille du batch impacte les performances finales et qu'un équilibre entre les performances finales et la taille du batch doit être trouvé (Goyal P., et al. 2017). Différentes tailles de batch sont utilisés pour résoudre le même problème d'optimisation à savoir la même base d'apprentissage et le même problème de classification. En particulier, le même taux d'apprentissage est utilisé pour chaque apprentissage. La Figure 83 montre que la taille du batch dans ce contexte a une influence importante sur les performances de classification : en effet, l'utilisation de grand batchs altère les performances.

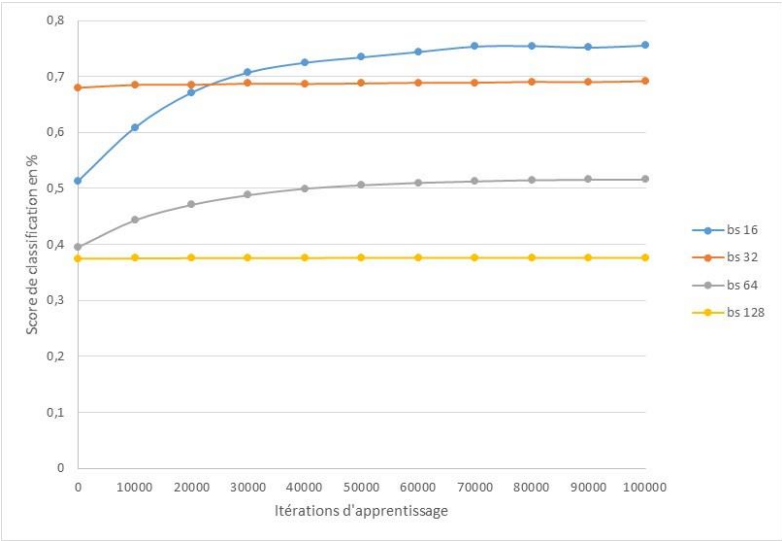

Figure 83: Graphique du score de validation au cours de l'apprentissage pour différentes valeurs de la taille de batch. Légende : bs=Batch Size

La Figure 84 présente la fonction de perte sur l'échantillon d'apprentissage durant les apprentissages précédents. L'optimisation a bien lieu puisque la fonction de perte diminue mais les vitesses de convergence varient suivant la taille du batch. La principale raison expliquant la relation entre la taille de batch et les performances est que l'estimation du gradient pour la descente bénéficie plus de l'utilisation de batch de petite taille. En effet, moyenner le gradient sur plusieurs exemples a tendance à faire diminuer la valeur du gradient moyen ce qui induit des petites mises à jour. En parallèle, l'analyse du temps de calcul présenté dans le Tableau 7 permet de mettre en avant le gain de temps observé avec l'utilisation de grands batchs. En effet, on peut noter que le temps de calcul n'évolue pas de manière linéaire avec la taille du batch. Et en particulier, l'efficacité de la méthode est calculée dans le Tableau 7 et est maximale pour une taille de batch de 128. En d'autres termes, pour une durée d'apprentissage fixée, un grand batch permet de traiter plus d'exemples et dans le cas d'un batch de 128, 30% plus d'exemples que dans le cas d'un batch de 16. Il est alors intéressant d'utiliser de grands batchs pour leur rapidité mais le taux d'apprentissage doit être adapté et optimisé conjointement à la taille du batch (Goyal P., et al. 2017).

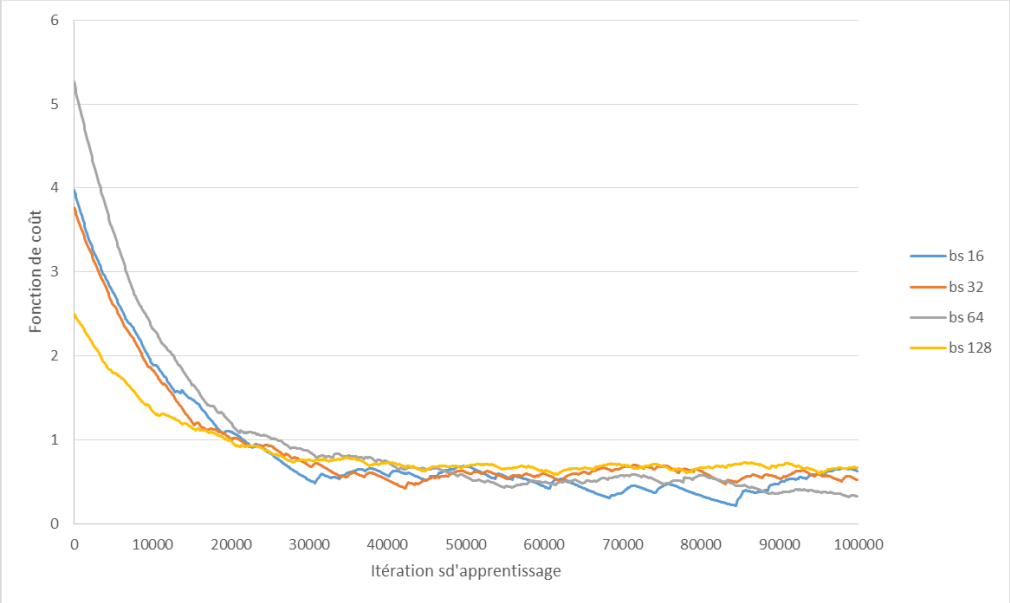

Figure 84: Graphique de la fonction de perte au cours du temps pour différentes valeurs de la taille de batch. Légende : bs=Batch Size

Taille du batch	16	32	64	128
Temps	28m	47m	1h28m	2h50
Efficacité (Exemples/s)	952	1135	1212	1255
Relatif à bs 16	0%	19%	27%	32%

Tableau 7: Temps de calcul pour différentes tailles de batch

Le choix des paramètres de l'algorithme d'optimisation change les performances finales mais la structure du CNN est supposée être le paramètre déterminant. La structure utilisée est une structure ConvNet qui a été paramétrée par certains hyperparamètres dans le paragraphe dédié (cf 5.3.4.3). Chaque hyper paramètre a un effet sur la structure et en particulier sur le nombre de poids à estimer. Les prochaines

analyses visent à évaluer les performances de différentes structures. La Figure 85 permet de visualiser les performances sur une base de test de 7 structures distinctes qui ont été obtenues en changeant différentes dimensions des couches cachées en fin de réseau : 32, 64, 128, 256, 512, 1024 et 2048. Les autres paramètres de structure ont été volontairement fixés à des valeurs par défauts de 3 pour la taille des convolutions et à 128 pour le nombre de convolutions. La Figure 85 permet de visualiser les performances des différentes structures en fonction du nombre de poids de la structure. Notons sur la figure de droite de la Figure 85 que les courbes des structures possédant moins de 128 unités sont confondues et que l'augmentation du nombre d'unités cachées améliore considérablement les performances. La complexité des modèles avec moins de 128 unités n'est pas suffisante pour le problème de classification. Au contraire, l'utilisation de grandes couches connectées avec 1024 et 2048 permettent d'obtenir la complexité nécessaire mais ces couches sont plus complexes à estimer puisqu'elles possèdent plus de poids à estimer et aboutissent à des performances plus faibles que des couches connectées plus petites (Tableau 8). Le problème de la détection de nuages ainsi formulé ne nécessite pas l'utilisation de grandes couches connectées à la différence de problème tel que Imagenet par exemple (Krizhevsky A., Sutskever I. et Hinton G. 2012). Il est important de noter que les couches connectées représentent près de 80% de l'ensemble des poids à estimer pour un réseau (Iandola F., et al. 2016).

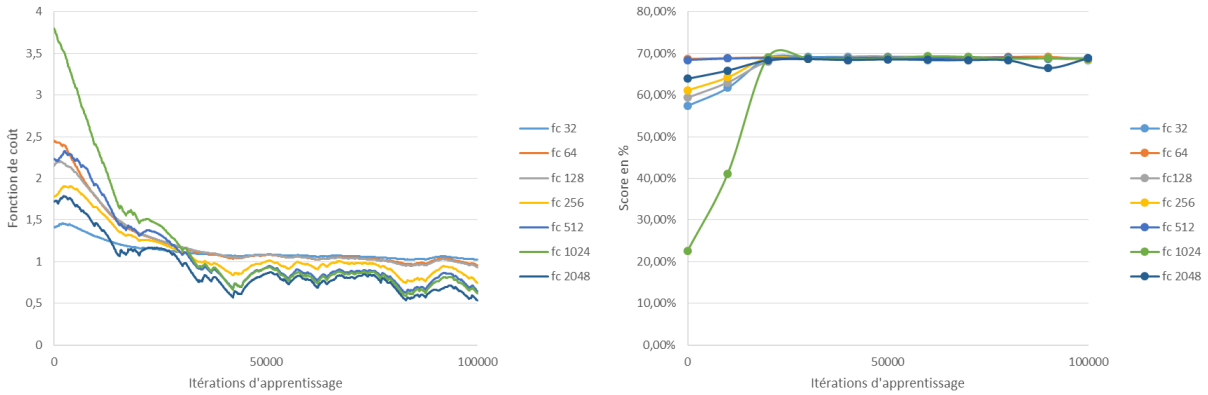

Figure 85: Erreur de validation pour différents structures en fonction du nombre de poids à estimer. Le nombre d'unités cachées sur les couches varie de 256 à 2048.

FC	32	64	128	256	512	1024	2048
Score	68,44%	68,50%	68,91%	69,01%	68,84%	68,78%	68,84%
Temps	47m	48m	47m	48m	50m	50m	1h05m

Tableau 8: Score final de validation pour chaque structure. FC est le nombre d'unités cachées dans les couches connectées.

Le bloc essentiel des CNNs est le bloc de convolutions qui possède également des paramètres à déterminer. Par exemple, la taille des convolutions utilisées est un de ces paramètres. De la même manière que précédemment, tous les autres paramètres de structures sont figés et la taille de convolutions peut prendre les valeurs 1, 3, 5, et 7. La Figure 86 montre les performances des différentes structures sur la détection de nuages. Tout d'abord, notons que l'influence de la taille des convolutions sur le nombre total de poids à estimer est négligeable comparé à la dimension des couches connectées de la figure précédente. Cependant, on peut également voir que l'augmentation de la taille des couches de

convolutions n'améliore que légèrement les performances finales sur la Figure 86 et le Tableau 9. Ce constat peut paraître contre intuitif mais il faut se rappeler que le problème qui est résolu est la classification de patch de taille 32x32. L'utilisation de taille de noyau de convolution trop important augmente les effets de bords lors d'une convolution sans augmenter les performances de détection. En particulier, sur une petite image, les effets de bords peuvent avoir une influence importance sur la sortie de la convolution et les images traitées en entrée des couches de convolution sont de taille 32×32 puis 16×16. De plus, on rappelle que ces constats sont similaires aux constats des auteurs de (Iandola F., et al. 2016) qui conseille d'utiliser des convolutions 3×3 pour limiter le nombre de poids à estimer sans dégrader les performances.

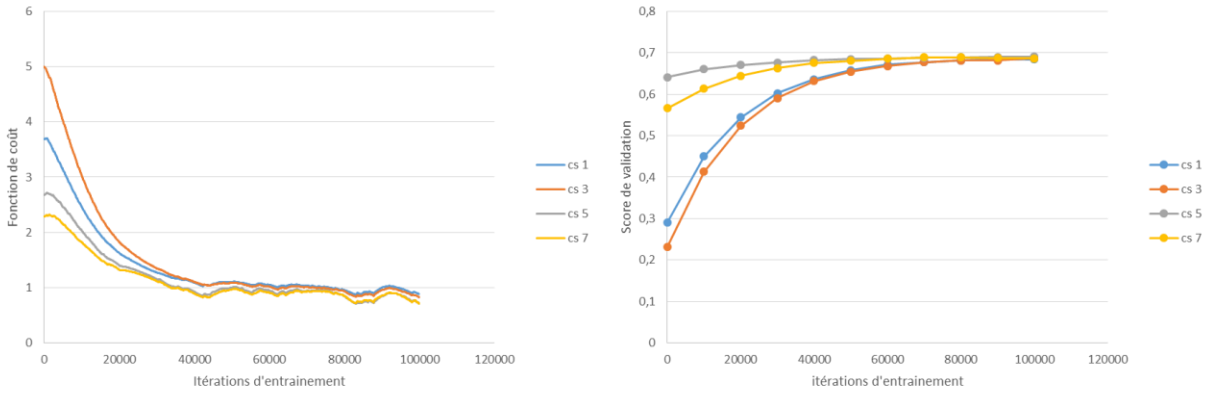

Figure 86: Erreur de validation pour différents structures en fonction du nombre de poids à estimer. La taille des noyaux des convolutions varie de 3 à 7.

Conv size	1	3	5	7
Score (%)	68.4	68.6	69.1	68.7
Temps	47m	49m	53m	57m

Tableau 9: Performances finales en fonction de la taille de convolution

Le dernier paramètre de structure est le nombre de filtres de convolutions dans les couches convolutives. Ici, ce paramètre varie de 16 à 256. De la même manière que pour la taille des convolutions, le nombre de filtres de convolutions a un effet négligeable sur le nombre total de poids à estimer. Pour cette étude, la taille des convolutions est fixée à 3 et le nombre d'unités cachées à 256. Les performances obtenues sur un échantillon de test sont mises en évidence sur la Figure 87. Le nombre de convolutions appliquées a un effet négligeable sur les performances finales. Cependant, il est intéressant de noter que la performance maximale est obtenue pour 128 convolutions par couche. Un nombre trop faible de convolution est insuffisant pour la classification et un nombre trop important de convolutions augmente le nombre de poids à estimer. De plus, le temps de calcul visible dans le Tableau 10 augmente considérablement avec le nombre de convolutions car les convolutions requièrent beaucoup de calculs. Un choix adapté du nombre de convolutions est donc nécessaire pour adapter la précision et le temps de calcul nécessaire pour l'apprentissage.

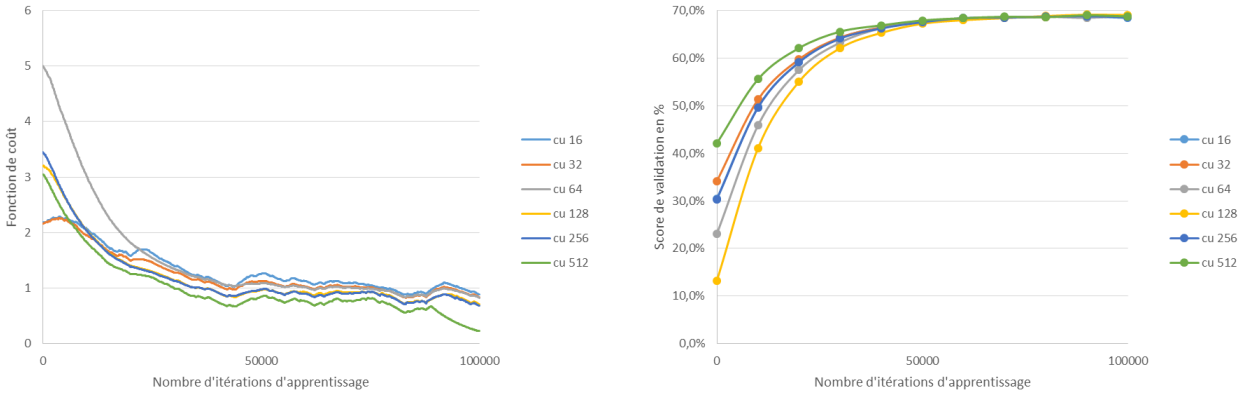

Figure 87: Erreur de validation pour différents structures en fonction du nombre de poids à estimer. Le nombre de convolutions varie de 32 à 256.

Conv units	16	32	64	128	256	512
Score (%)	68.4	68.7	68.8	68.9	68.6	68.6
Temps	44m	46m	49m	57m	1h15	2h11

Tableau 10: Performances en fonction du nombre de convolutions

5.4.2.1.2 Discussions

Tous ces résultats sont spécifiques au cas d'utilisation de la détection de nuages, à l'analyse de petites imagettes (32x32) satellite et à la base de test utilisée. La plupart de ces analyses devront être reconduites pour d'autres cas d'utilisation. De la même manière, les analyses de structure ont été faites avec des paramètres d'optimisation fixes : une analyse plus robuste consisterait à trouver les meilleurs paramètres d'optimisation pour chaque variation de structure. En effet, des structures possédant de grands nombres de paramètres ou beaucoup de convolutions ne s'optimisent pas de la même manière que les autres. De plus, la détection de nuages est un problème simple comparé à la classification d'images quelconques telles que celles considérées dans Imagenet. Les cas difficiles de classification de nuages sont des cas statistiquement rares. La « simplicité » du problème permet d'expliquer que la taille des réseaux utilisés est relativement faible comparativement à celles des réseaux habituellement utilisés dans le domaine. Pour conclure cette phase d'analyse de structure, la paramétrisation de la structure n'a permis que de tester qu'un petit nombre de structures et cette analyse devrait être étendue aux récentes familles de structures telles que les ResNets, les Inceptions, … Notons cependant que ces familles ont été conçues pour étudier des grandes images d'une largeur excédant une centaine de pixels et qu'une adaptation est nécessaire avant leur utilisation pour l'analyse de petites images comme celles utilisées ici.

5.4.2.2 Analyse qualitative

5.4.2.2.1 Résultats qualitatifs

L'analyse des résultats qualitatifs correspond à l'analyse morphologique des résultats de prédiction. Pour la détection de nuages, cette analyse est réalisée à partir de l'étude d'un panel d'images correspondant à des cas faciles et difficiles. La Figure 88 montre les résultats de la détection de nuages avec un réseau convolutionnel en utilisant la méthodologie de la fenêtre glissante expliquée dans la partie sur la

modélisation 5.2.2. La Figure 88 présente les performances de prédiction sur des images qui correspondent à des cas faciles correspondant à des nuages épais et blancs. Les performances de détection sont satisfaisantes et les principaux nuages peuvent être identifiés. Une différence est tout de même observée et le manque de précision du CNN est expliquée par l'utilisation de patch de taille 32×32.

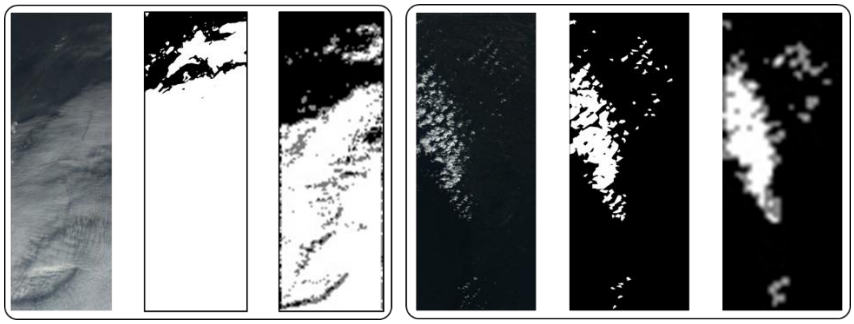

Figure 88: Exemples de prédictions du CNN pour la détection de nuages. Ordre : de gauche à droite, image, masque, prédictions

Pour compléter les résultats de la Figure 88, on considère des images contenant d'autres artéfacts causant des mauvaises détections comme la neige, les sables, … La Figure 89 présente la réponse du CNN à certains de ces artefacts : les principaux artefacts restent des cas difficiles même pour le CNN à cette résolution. La neige sur la deuxième imagette trompe le CNN et l'œil humain par la même occasion. Les sables côtiers de la première image engendrent également une mauvaise classification et enfin, les nuages extrêmement fins de la dernière image continuent de tromper le détecteur de nuages. La performance sur les cas difficiles n'a pas été améliorée mais la précision du classifieur sur les cas faciles a été améliorée par l'utilisation du CNN.

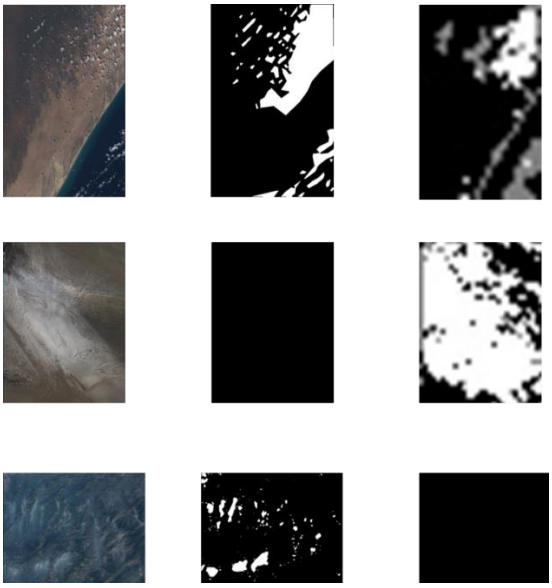

Figure 89: Exemples de mauvaises prédictions du CNN pour la détection de nuages. Ordre : de gauche à droite, image, masque, prédictions

La complexité de la détection de nuages a souvent été un frein pour les analyses et l'avantage de l'apprentissage profond est de pouvoir s'adapter à des cas difficiles où des descripteurs ne sont pas connus. Dans un second temps, nous considérons la classification de terrain qui est intéressante pour

tester les algorithmes de classification construits à partir de CNN. Cependant, comme il a été indiqué dans la description du cas d'étude, la classification de terrain en utilisant notre vérité terrain risque d'être difficile à cause de la résolution des images. Les premiers tests qualitatifs ont simplement consisté à analyser les résultats des classifieurs sur des images tests. Les connaissances disponibles sur le sujet, en particulier, sur l'utilisation de la vérité terrain, sont limitées et n'ont pas permis d'identifier des cas typiques d'erreurs de classifieurs. Les classifications obtenues dans la suite de cette partie sont issues de l'analyse empirique d'un grand nombre d'images tests. Le classifieur utilisé dans cette partie est issu des familles précédentes des ConvNets où la taille des convolutions a été fixée à 3, le nombre de convolution à 64 et le nombre d'unités cachées à 512. Ces paramètres ont été sélectionnés avec la même analyse que celle qui a été présentée pour la détection de nuages. La Figure 90 montre un exemple de classification où l'on peut voir que des nuages sont détectés mais que les classes de sols présentes sur l'image sont mélangées avec la classe de champs en friche. En réalité, un grand nombre de classes sont regroupées cette classe de champs en friche ce qui explique les faibles performances sur la base de données d'entrainement.

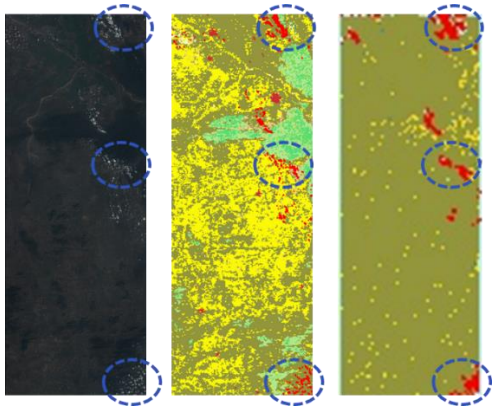

Figure 90: Exemple de prédiction de couverture pour une image.

Certains types de terrain comme l'eau sont détectés par le classifieur comme le montre la Figure 91. Il faut noter que la détection d'eau est comme la détection de nuages simple a priori mais l'ajout des déformations liées aux conditions d'acquisition perturbe les détecteurs les plus simples. Par exemple, on peut remarquer un contraste faible sur les détections de la Figure 91 et l'eau est tout de même détectée dans des contextes côtiers plus classiques comme on peut le voir dans la Figure 92.

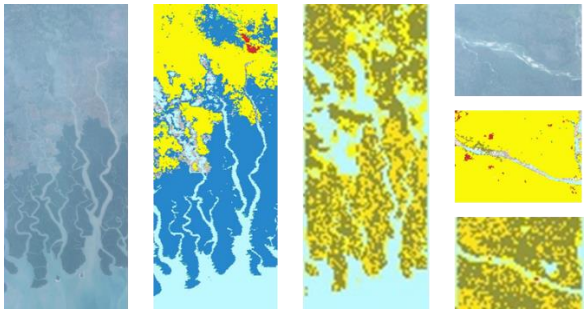

Figure 91: Exemple de détection d'eaux. Gauche: de gauche à droite, image, vérité terrain, et classification. Droite: de haut en bas, image, vérité terrain, classification

En réalité, la détection de couverts a posé un autre problème au classifieur : il devait au fur et à mesure de son apprentissage établir la faisabilité de sa classification. Il n'y a aucune certitude que le problème de la classification des sols tel qu'il est posé à partir de patches d'images album est réalisable. De même, plus tard dans cette analyse, l'interprétation humaine des patches sera évaluée. L'idée est que l'algorithme prenne la meilleure décision compte tenu de l'information à disposition. Ceci explique pourquoi l'algorithme a fusionné des classes car il n'a pas assez d'information pour les classifier correctement. De manière pratique, cela se matérialise par le fait que l'algorithme ne prédit qu'un ensemble restreint de classes comme on peut le voir dans la Figure 92 où seulement 4 classes prédites apparaissent à savoir champs en friche (vert), champs d'agriculture (jaune), et prairies (beige). Les différentes images de prédiction permettent d'observer que les terrains contenus sur le sol sont majoritairement classés en champs friche ou agricoles (Figure 90 et Figure 92).

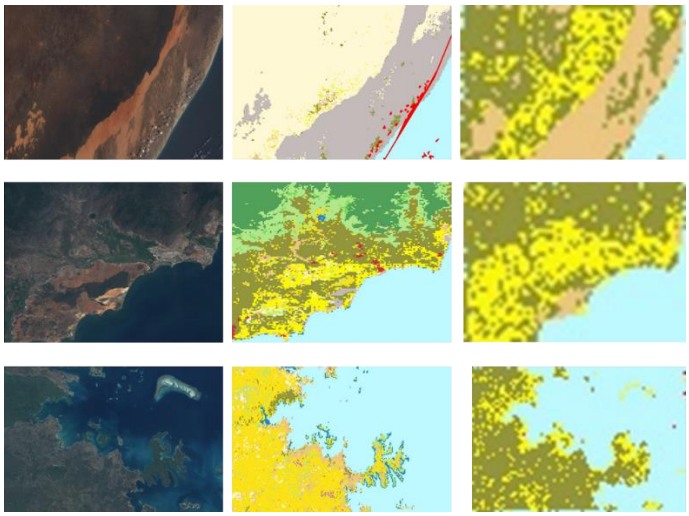

Figure 92: Exemple de classification de terrain sur différents types de paysage côtiers.

Comme il vient d'être expliqué, les faibles performances du classifieur de terrain peuvent être liées au fait que l'information à disposition sous la forme d'un patch d'image album 32×32 d'image album n'est pas suffisante pour effectuer la classification. Il nous a semblé intéressant d'évaluer si l'interprétation humaine peut surpasser la performance de l'algorithme automatique sur ce cas d'utilisation.

Cependant, les premiers tests se sont avérés très compliqués car l'interprétation de patches 32x32 d'image satellite est complexe. De plus, il n'existe pas de définition exacte des classes : il n'y a priori aucune information mis à part la base de données à disposition. La première expérience a consisté à classifier des patches d'images à la main. Les classes sur lesquelles les performances du classifieur sont satisfaisantes correspondent aux classes que l'analyse humaine peut récupérer : à savoir, les classes d'eau, de nuages, et de territoires « arides ». Cependant, pour les autres classes qui appartiennent aux classes de sols (non eau, et non nuages), la distinction entre classes est très difficile comme peut le démontrer quelques exemples de la Figure 94 où l'identification de la classe « forêt » ne peut pas être réalisée à ce niveau de résolution. De la même manière, une analyse rapide de la base de données permet de mettre en évidence que des images extrêmement similaires peuvent avoir des classes de terrain extrêmement différentes. La complexité du problème de classification des sols à cette résolution est alors prouvée et peut expliquer la faible performance du classifieur appris ainsi que le comportement macroscopique de celui-ci.

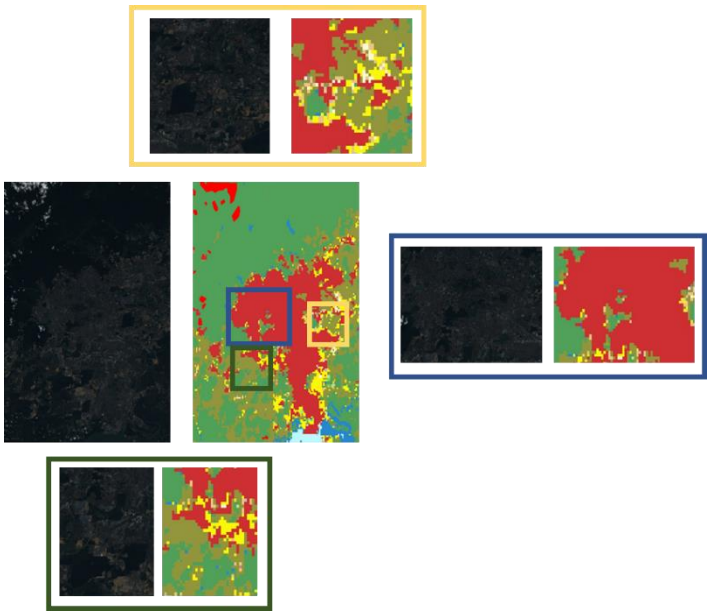

Figure 93: Exemple de patchs issus d'une même image.

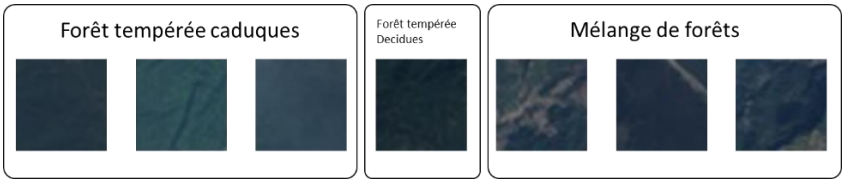

Figure 94: Complexité de la classification des types de forêts.

5.4.2.2.2 Discussions

Les dernières expériences associées aux Figure 89 et Figure 92 ont permis de mettre en évidence les limitations de l'approche actuelle. Cependant, le comportement de ce classifieur actuel pourrait être analysé encore plus en profondeur. Seule la classe prédite a été analysée mais le classifieur associe un degré de confiance à chaque classe pour chaque prédiction qui pourrait être analysé et utilisé pour mieux faire la classification. De manière plus générale, le classifieur semble avant tout souffrir du manque d'information : en effet, l'apprentissage profond nous assure que s'il était possible de le faire, le réseau final le ferait puisqu'il a les degrés de liberté nécessaires. Diverses approches peuvent permettre d'augmenter l'information à disposition. Une taille de patch plus grande pourrait être utilisée et de manière plus générale, des prédictions avec différentes tailles de patchs pourraient être combinées (Farabet C., et al. 2013). Pour conclure, il semblerait que la résolution actuelle des images albums ne soit pas suffisante pour faire de la classification des sols et l'augmenter en utilisant d'autres images est un axe prometteur de recherche.

5.5 BILAN SUR L'APPRENTISSAGE PROFOND POUR LA TELEDETECTION

5.5.1 Bilan

Le bilan principal de cette partie est que l'utilisation de l'apprentissage profond a permis d'atteindre et de dépasser les performances atteintes par les techniques classiques d'apprentissage du niveau de l'état de l'art et que l'application de la même technique sur un cas d'utilisation similaire a permis d'obtenir des résultats encourageants. L'étude a porté sur l'étude et la construction d'une structure adaptée pour faire de la classification de nuages et de la classification de sols. La complexité des structures et des algorithmes a nécessité que certains choix soient faits sur la base d'une étude bibliographique car toutes les configurations ne pouvaient pas être testées dans le cadre de cette thèse. Le monde de l'apprentissage profond est basé sur des observations empiriques et le mode opératoire utilisé ici est donc en accord avec cette vision. Les performances finales obtenues ont permis de valider les premiers choix de structure réalisés mais des expériences supplémentaires sont nécessaires pour tester d'autres configurations que celles choisies : par exemple, un algorithme sans moments, différentes formulations du taux d'apprentissage, …. De même, il a été mis en évidence, en particulier, pour le cas de la classification des sols que de nouvelles métriques doivent être mis en place comme la Top-3 accuracy[8] actuellement utilisé dans d'autres travaux (Deng J., et al. 2009).

De manière plus générale, la paramétrisation des structures doit être revue de manière à étudier un plus grand nombre de structures possibles et en particulier, une topologie plus variée. Il a déjà été cité que certaines familles d'architectures ne sont pas adaptées pour l'étude d'imagettes (patches de taille 32×32) mais elles doivent tout de même être adaptées et inclues dans les prochaines études de structure.

5.5.2 Perspectives

La recherche de meilleures structures n'aboutira pas forcément à un grand bouleversement des performances : la principale limite de l'apprentissage est souvent la formulation du problème. La formulation utilisée pour la détection de nuages et la classification de sols se ramène à une décision prise pour chaque pixel indépendamment des décisions associées aux autres pixels. Une autre formulation consisterait à formuler le problème comme un problème de labélisation sémantique où une classe est attribuée à chaque pixel pour toute l'image. Cette formulation a déjà été évoquée auparavant mais elle fut abandonnée car les approches ou modèles actuels ne semblaient pas avoir résolu le problème. Mais le domaine de l'apprentissage profond avance vite et l'évolution naturelle des approches proposées consiste à appliquer des réseaux de labélisation sémantique (Ronneberg O., Fischer P. et Brox T. 2015). La même analyse que celle réalisée dans cette partie sera nécessaire pour déterminer les meilleurs paramètres de structures pour ce nouveau type de problème.

[8] Prédiction des trois classes les plus probables

CONCLUSION GENERALE

Chapitre 6 CONCLUSION GÉNÉRALE

CONTEXTE

Grâce à leur couverture spatiale et à leur fréquence d'acquisition, les images satellitaires présentent un grand intérêt pour l'analyse de la surface de la Terre. La télédétection, autrefois appliquée à l'étude d'une seule image, s'est progressivement tournée vers l'analyse d'un grand nombre d'images. Cette thèse s'est focalisée sur les techniques d'extraction de contenu appliquées à des volumes de données massifs. En effet, avec l'arrivée de nouvelles images, par exemple issues du programme européen Copernicus, un nouveau cap est franchi en termes de quantité de données à exploiter. De nouveaux services basés sur l'extraction d'informations de ces images satellitaires sont donc en cours d'étude. En parallèle, une révolution technologique s'est opérée dans le monde de l'informatique offrant de nouveaux outils performants aux utilisateurs pour ces analyses. Une grande puissance de calcul est aujourd'hui accessible au grand public grâce au cloud computing. Les technologies d'apprentissage automatique se sont développées massivement ces dernières années et sont aujourd'hui appliquées avec succès à un grand nombre de domaines d'application.

Afin d'adapter les chaînes de traitement d'images satellitaires à ce changement, un nouveau modèle de classification est introduit afin d'optimiser les calculs distribués sur un réseau de machines. L'apparition récente des techniques d'apprentissage profond a par la suite orienté l'étude vers leur application aux images satellitaires. Par conséquent, une nouvelle méthodologie de résolution des problèmes classiques de traitement d'images spatiales est alors proposée. L'étude présentée dans ce manuscrit et résumée ci-après a permis de formuler les conclusions suivantes.

CONCLUSIONS

Le Chapitre 3 a permis de lister les principales applications de traitement d'images spatiales. En particulier, deux cas d'utilisation sont étudiés en profondeur : la détection de nuages et la classification des sols. Ce chapitre introduit également les nouveaux outils informatiques utilisés pour les résoudre et la nécessité de définir de nouvelles méthodologies pour leur utilisation. Parmi les modèles de classification, les travaux se sont concentrés sur l'algorithme du Boosting. Les performances du Boosting ont déjà été prouvées par ailleurs et la simplicité de sa structure était propice aux adaptations pour le passage à l'échelle. Le Chapitre 4 propose alors une revue des adaptations distribuées existantes pour cet algorithme. Cette revue a permis d'identifier les étapes performantes des implantations distribuées, leurs faiblesses et les métriques utilisées pour leur comparaison. Ces analyses ont conduit à la conception d'un nouvel algorithme de Boosting distribué dans le Chapitre 4. Celle-ci s'est concentrée sur l'optimisation des propriétés de passage à l'échelle en maintenant la structure performante du Boosting. Les expériences montrent que l'algorithme proposé rivalise avec l'état de l'art en termes de propriétés de passage à l'échelle. De plus, les performances de classification surpassent celles des approches traditionnelles appliquées à la détection de nuages.

En parallèle de ces travaux, les techniques d'apprentissage profond et des outils associés se sont développés de manière exponentielle. Le Chapitre 3 propose un résumé des différents travaux réalisés sur des cas d'utilisation similaires à ceux de cette thèse. Il est alors montré que ces techniques bouleversent les approches traditionnelles de résolution des problèmes de traitement d'images. Le Chapitre 5 propose

une nouvelle méthode de résolution par apprentissage profond de problèmes classiques de traitement d'images satellitaires que sont la détection de nuages et la classification des sols. Les méthodes d'apprentissage profond sont aujourd'hui principalement appliquées aux images issues du Web et nécessitent d'être adaptées aux spécificités des images satellitaires. La structure et la chaîne de traitement proposées sont issues de l'application des méthodes performantes d'apprentissage profond modifiées pour les images spatiales. Les résultats obtenus avec cette nouvelle méthodologie et les différentes structures neuronales dépassent les performances obtenues avec les méthodes traditionnelles.

CONTRIBUTIONS APPLICATIVES

Dans les travaux réalisés, les différentes méthodes développées permettent de résoudre un problème pertinent du domaine spatial : la classification d'images satellitaires. Les chaînes de traitement actuelles sont désormais dépassées et doivent évoluer car elles ne répondent pas à la demande de quantité de données à traiter et de la qualité d'information à extraire.

La mise en place des techniques présentées dans cette thèse possède une grande composante informatique et nécessite la maîtrise d'un grand nombre de technologies. En effet, le déploiement des calculs de l'algorithme de Boosting a nécessité une maîtrise des méthodes de déploiement, de gestion et d'orchestration de clusters de machines dans le cloud. En outre, l'utilisation d'outils dédiés comme Spark fut nécessaire pour faciliter l'exécution distribuée des algorithmes. Enfin, la gestion des données dans un système de fichiers distribué tel que le HDFS diffère de la gestion habituelle locale. Les outils et méthodes liés à l'apprentissage profond sont également apparus en majorité durant la thèse. A noter que la plupart de ces technologies ont connu une croissance exponentielle durant cette période pour atteindre la maturité qu'elles connaissent aujourd'hui. La mise en pratique de ces algorithmes a donc permis le développement de savoir-faire essentiels à la conception des segments sols de nouvelle génération. La suite du projet OCE de l'IRT au sein duquel a été réalisée la thèse a pour but de développer une plateforme pour la conception des futurs segments sols. Les algorithmes développés ainsi que les savoir-faire associés seront donc intégrés à cette plateforme, nommée CITADEL[9]. Elle consiste en une suite logicielle dont le but est d'extraire de l'information et traiter des bases de données massives pour accélérer le développement d'applications spatiales. En particulier, CITADEL incorpore un nombre important d'outils dédiés à l'utilisation du cloud et de l'apprentissage automatique qui sont développés grâce à des cas d'utilisation concrets. Ainsi, ceux étudiés dans cette thèse, la détection de nuages et la classification des sols dans des images SPOT 6, ont permis de développer un nombre important de services désormais intégrés à cette plateforme.

PERSPECTIVES

L'étude menée dans ce manuscrit a soulevé plusieurs perspectives de recherche qui sont décrites dans les paragraphes suivant.

Passage à l'échelle

Tout d'abord, des simulations sur des jeux de données plus massifs que ceux déjà traités devront être intégrées par la suite. Des expériences de passage à l'échelle appliquées à l'apprentissage profond seraient

[9] CITADEL *Computing Infrastructure for Technology Assessment, Development and EvaLuation*

utiles pour compléter le Chapitre 5 afin d'étudier le comportement de telles techniques vis-à-vis de la puissance de calcul réquisitionnée pour l'apprentissage.

Segmentation sémantique

Le modèle actuel basé sur des prédictions indépendantes par pixel ne permet pas de capturer les phénomènes complexes mis en jeu. Par exemple, ne prenant pas en compte les formes des nuages prédits, il ne peut pas les utiliser pour déceler les mauvaises détections. Il serait intéressant de considérer une prédiction structurée où les prédictions des pixels voisins sont prises en compte sous la forme d'un modèle proche de celui utilisé pour la segmentation d'images. Il reste cependant à identifier la modification nécessaire à la procédure distribuée de Boosting pour l'adapter à cette nouvelle formulation. Des modifications ont déjà été identifiées dans le Chapitre 4 comme celle du Gradient Boosting et doivent être intégrées en priorité avant celle de la prédiction structurée. Une modification de la modélisation actuelle du problème d'apprentissage profond est également nécessaire pour incorporer ce type de prédiction. Cette modification changerait alors la topologie et les sorties du modèle et impliquerait donc l'utilisation de nouvelles structures neuronales qu'il reste encore à déterminer.

Optimisation de structure

Le problème d'apprentissage profond décrit dans le Chapitre 5 fait intervenir plusieurs hyperparamètres de structure choisis par validation croisée durant les diverses expériences. La mise en place de procédures automatisées de choix de ces paramètres, dans le cadre des algorithmes présentés jusqu'ici, reste à étudier. Les hyperparamètres considérés dans cette thèse ont été déterminés en fixant une topologie de structure neuronale : la structure ConvNet. Or une procédure automatique ne doit pas dépendre d'un patron de structure mais trouver la meilleure structure pour le problème cible. A ce sujet, une approche basée sur une recherche par algorithme génétique peut être envisagée. Cette optimisation sera possible dès lors qu'une paramétrisation performante des structures neuronales sera disponible.

Combinaison des approches

Pour finir, une méthode permettant de combiner les deux approches étudiées reste à définir. En effet, la conception de l'algorithme de Boosting distribué a été optimisée pour le passage à l'échelle. Cependant, les expériences ont montré que ses performances dépendent des descripteurs utilisés en entrée. En parallèle, l'apprentissage profond permet d'estimer des descripteurs performants et spécifiques à une tâche. La complémentarité des deux approches est alors indéniable et pourrait permettre d'apprendre des descripteurs performants sur une grande quantité de données. En effet, les descripteurs pourraient être estimés sur une petite base de données par apprentissage profond et la fonction de décision globale serait calculée sur une plus grande quantité de données grâce au passage à l'échelle du Boosting distribué.

Applications

La formulation des modèles de classification étudiées dans cette thèse peut présenter un intérêt dans d'autres contextes par exemple la détection d'objets comme les avions, les bateaux ou les bâtiments dans des images satellitaires. En effet, la détection d'objets peut être simplifiée à un problème de classification binaire en considérant la classe objet et la classe arrière-plan. Les algorithmes de Boosting distribué et d'apprentissage profond pourraient alors être appliqués de manière similaire. De plus, l'utilisation d'images satellitaires de résolution spatiale supérieure devrait permettre d'apporter des informations complémentaires pour la détection de nuages et la classification des sols sans modifier les méthodes.

Enfin, l'algorithme de Boosting distribué du Chapitre 4 pourrait être appliqué à des cas d'utilisation d'apprentissage plus traditionnels comme le traitement de textes. En effet, un grand nombre d'applications nécessite de traiter des volumes massifs de données en possédant des descripteurs déjà performants et pourrait alors bénéficier de l'utilisation de l'algorithme de Boosting distribué.

BIBLIOGRAPHIE

Abadi M., Agarwal A., Barham P., Brevdo E., Zhifeng C., Citro C., Corrado G., et al. 2016. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems." *arXiv preprint arXiv:1603.04467.*

Abualkibash M., El Sayed A., and Mahmood A. 2013. "Highly scalable, parallel and distributed Adaboost algorithm using light weight threads and web services on a network of multicores machines." *International Journal of Distributed and Parallel Systems (IJDPS)* 29-40.

Acchanta R., Shaji A., Smith K., Lucchi A., Fua P., and Susstrunk S. 2012. "SLIC superpixels compared to state-of-the-art superpixel methods." *IEEE trans. on Pattern Analysis and Machine Intelligence* 2274-2282.

Audebert N., Le Saux B., and Lefevre S. 2016. "How useful is region-based classification of remote sensing images in a deep learning framework?" *Proc. of 2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS).* Beijing. 5091-5094.

Bengio Y., Courville A., and Vincent P. 2013. "Representation learning: A review and new perspectives." *IEEE trans. on Pattern Analysis and Machine Intelligence.* 1798-1828.

Bergstra J., and Bengio Y. 2012. "Random search for hyper-parameter optimization." *Journal of Machine Learning Research* 281-305.

Bottou L. 2012. "Stochastic gradient descent tricks." In *Neural Networks: Tricks of the trade*, 421-436. Springer.

Bradley J., and Schapire R. 2007. "Filterboost: Regression and classification on large datasets." *Adv. in Neural Processing Information Systems.* Vancouver. 185-192.

Breiman L. 2001. "Random forests." *Machine Learning* 5-32.

Breiman L., Friedman J., Olsen, and Stone. 1984. *Classification and Regression trees.* CRC Press.

Browton P. 2014. "A global land cover climatology using MODIS data." *Journal of Applied Meteorology and Climatology* 1593-1605.

Canziani A., Paszke A., and Culurciello E. 2016. "An Analysis of Deep Neural Network Models for Practical Applications." *arXiv Preprint arXiv:1605.07678.* arXiv.

Chandran A., and Christy J. 2015. "A Survey of Cloud Detection Techniques For Satellite Images." *International Research Journal of Engineering and Technology* 2486-2490.

Chen T., and Guestrin C. 2016. "Xgboost: A scalable tree boosting system." *Proc. of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* San Francisco: ACM. 785-794.

Chollet, F. 2016. "Xception: Deep Learning with Depthwise Separable Convolutions." *arXiv preprint arXiv:1610.02357.*

Collins M., Schapire R., and Singer Y. 2002. "Logistic regression, AdaBoost and Bregman distances." *Machine Learning* 253-285.

Comaniciu D., and Meer P. 2002. "Mean shift: A robust approach toward feature space analysis." *IEEE trans. on Pattern Analysis and Machine Intelligence* 603-6019.

Cooper J., and Reyzin L. 2014. "Improved algorithms for distributed boosting." *NIPS Workshop on Distributed Machine Learning and Matrix Computations.* Montreal.

Criminisi A., Shotton J., and Konukoglu E. 2012. "Decision Forests for Classication, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning." *Foundations and Trends in Computer Graphics and Vision* 81-227.

Deng J., Dong W., Socher R., Li L., Li K., and Fei-Fei L. 2009. "Imagenet: A large-scale hierarchical image database." *Proc. of IEEE International Conference on Computer Vision and Pattern Recognition.* Miami, USA. 248-255.

Donahue J., Jia Y., Vinyals O., Hoffman J., Zhang N., Tzeng E., and Darell T. 2014. "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition." *Proc. of IEEE Internation Conference on Machine Learning.* Beijing, China: IEEE. 647-655.

Dubout C., and Fleuret F. 2014. "Adaptive Sampling for Large Scale Boosting." *Journal of Machine Learning Research* 1431-1453.

Farabet C., Couprie C., Najman L., and LeCun Y. 2013. "Learning hierarchical features for scene labeling." *IEEE trans. on Pattern Analysis and Machine Intelligence.* 1915-1929.

Freund Y., and Schapire R. 1996. "Experiments with a new boosting algorithm." *Proc. of International Conference on Machine Learning.* Bari, Italia. 148-156.

Friedman J., Hastie T., and Tibishirani R. 2001. *The elements of statistical learning.* Berlin: Springer series in statistics.

Fukushima, K. 1988. "Neocognitron: A hierarchical neural network capable of visual pattern recognition." *Neural networks* 119-130.

Geurts P., Ernst D., and Wehenkel L. 2006. "Extremely Randomized Trees." *Machine Learning* 3-42.

Goodfellow I., Bengio Y., and Courville A. 2016. *Deep learning.* MIT Press.

Goodfellow, I. 2016. "NIPS 2016 Tutorial: Generative Adversarial Networks." *arXiv Preprint arXiv:1701.00160.*

Goyal P., dollar P., Girshick R., Noordhuis P., Wesolowski L., Kyrola A., Tulloch A., Jia Y., and He K. 2017. "Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour." *arXiv preprint arXiv:1706.02677.*

Graham, B. 2014. "Spatially-sparse convolutional neural networks." *arXiv preprint arXiv:1409.6070.*

Hagolle O., Huc M., Villa Pascual D., and Dedieu G. 2010. "A multi-temporal method for cloud detection, applied to FORMOSAT-2, VENµS, LANDSAT and SENTINEL-2 images." *Remote Sensing of Environment* 1747-1755.

He K., Zhang X., Ren S., and Sun J. 2016. "Deep residual learning for image recognition." *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition.* Washington, USA. 770-778.

Hollstein A., Segl K., Guanter L., Brell M., and Enseco M. 2016. "Ready-to-use methods for the detection of Clouds, Cirrus, Snow, Shadow, Water and Clear Sky pixels in Sentinel-2 MSI images." *Remote Sensing* 666.

Howard, A. 2014. "Some Improvements on Deep Convolutional Neural Network Based Image Classification." *Proc. of International Conference on Machine Learning .* Beijing.

Huang C., Davis L., and Townshend J. 2002. "An assessment of support vector machines for land cover classification." *International Journal of Remote Sensing* 725-749.

Iandola F., Han S., Moskewicz M., Ashraf K., Dally W., and Keutzer K. 2016. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size." *arXiv Preprint arXiv:1602.07360.*

Ienco D., Gaetano R., Dupaquier C., and Maurel P. 2017. "Land Cover Classification via Multi-temporal Spatial Data by Recurrent Neural Networks." *arXiv Preprint arXiv:1704.04055.*

Jedlovec, G. 2009. "Automated detection of clouds in satellite imagery." *Adv. in Geoscience and Remote Sensing.* InTech.

Jovanovic V., and Risojevic V. 2014. "Evaluation of bag-of-colors descriptor for land use classification." *22nd Telecommunications Forum (Telfor).* Belgrade, Serbia: IEEE. 889-892.

Kalal Z., Matas J., and Mikolajczyk K. 2008. "Weighted sampling for large-scale boosting." *Proc. of British Machine Vision Conference.* London.

Karpathy, A. 2016. "Connecting Images and Natural Language." PhD Thesis.

Kingma D., and Ba J. 2015. "Adam: A method for stochastic optimization." *Proc. of International Conference on Learning Representations.* San Diego, USA.

Krizhevsky A., Sutskever I., and Hinton G. 2012. "Imagenet classification with deep convolutional neural networks." *Adv. in Neural Information Processing Systems.* Lake Tahoe, USA. 1097-1105.

Lassale, P. 2015. "Etude du passage à l'échelle des algorithmes de segmentation et de classification en télédétection pour le traitement de volumes massifs de données." PhD thesis, Toulouse.

Latry C., Panem C., and Dejean P. 2007. "Cloud detection with SVM technique." *Proc. of IEEE International Geoscience and Remote Sensing Symposium.* Barcelona, Spain. 448-451.

Lazarevic A., and Obradovic Z. 2001. "The distributed boosting algorithm." *Proc. of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* San Francisco: ACM. 311-316.

Le Goff M., Tourneret J-Y, Wendt H., Ortner M., and Spigai M. 2016. "Distributed boosting for cloud detection." *Proc. of 2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS).* Beijing: IEEE. 2626-2629.

Le Goff M., Tourneret J-Y, Wendt W., Ortner M., and Spigai M. 2017. "Deep Learning for Cloud Detection." *Proc. of International Conference on Pattern Recognition Systems (ICPRS).* Madrid.

Le Hégarat-Mascle S., and André C. 2009. "Use of Markov random fields for automatic cloud/shadow detection on high resolution optical images." *Journal of Photogrammetry and Remote Sensing* 351-366.

LeCun Y., Bottou L., Bengio Y., and Haffner P. 1998. "Gradient-based learning applied to document recognition." *Proc. of the IEEE* 2278-2324.

Li M., Zhou L., Yang Z., Li A., Xia F., Andersen D., and Smola A. 2013. "Parameter server for distributed machine learning." *Big Learning NIPS Workshop.* Lake Tahoe. 2.

Lin M., Qjang C., and Schuicheng Y. 2013. "Network in network." *arXiv preprint arXiv:1312.4400.*

Long J., Shelhamer E., and Darell T. 2015. "Fully convolutional networks for semantic segmentation." *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition.* Boston. 3431-3440.

Luus F., Salmon B., Van der Bergh F., and Maharaj B. 2015. "Multiview deep learning for land-use classification." *IEEE Geoscience and Remote Sensing Letters* 2448-2452.

Mnih, V. 2013. "Machine Learning for Aerial Image Labeling." PhD Thesis, Toronto.

Natekin A., and Knoll A. 2013. "Gradient boosting machines, a tutorial." *Frontier in Neurorobotics.*

Negrel R., Picard D., and Gosselin P. 2014. "Evaluation of second-order visual features for land-use classification." *12th International Workshop on Content-Based Multimedia Indexing (CBMI).* Klagenfurt, Austria: IEEE. 1-5.

Noureldin L., Ayman N., Onsi M., and El Saban H. 2012. "Spatial cloud detection and retrieval system for satellite images." *International Journal of Advanced Computer Science and Applications (IJACSA).*

Panda B., Herbach J., Bayardo R., and Basu S. 2009. "PLANET: Massively Parallel Learning of Tree Ensembles with MapReduce." *Proc. of the 35th International Conference on Very Large Data Bases (VLDB-2009).* Lyon: VLDB Endowment. 1426-1437.

Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. "You only look once: Unified real-time object detection." *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition.* Las Vegas, USA: IEEE. 779-788.

Ren S., He K., Girshick R., and Sun J. 2015. "Faster R-CNN: Towards real-time object detection with region proposal networks." *Adv. in Neural Information Processing Systems* 91-99.

Rodes, I. 2016. "Estimation de l'occupation des sols à grande échelle pour l'exploitation d'images d'observation de la Terre à hautes résolutions spatiale, spectrale et temporelle." PhD Thesis, Toulouse.

Ronneberg O., Fischer P., and Brox T. 2015. "U-net: Convolutional networks for biomedical image segmentation." *Proc. of International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI).* Munich, Germany. 234-241.

Schapire R., Freund Y., Bartlett P., and Lee W. 1998. "Boosting the margin: A new explanation for the effectiveness of voting methods." *Annals of Statistics* 1651-1686.

Schapire, R., and Y. Singer. 1999. "Improved boosting algorithms using confidence-rated predictions." *Machine Learning* 297-336.

Sermanet P., Eigen D., Zhang X., Mathieu M., Fergus R., and LeCun Y. 2014. "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks." *Proc. of International Conference on Learning Representations.* Banff, Canda.

Shi M., Xie F., Zi Y., and Yin J. 2016. "Cloud detection of remote sensing images by deep learning." *Proc. of IEEE International Geoscience and Remote Sensing Symposium (IGARSS).* Beijing, China: IEEE. 701-704.

Simonyan K., and Zisserman A. 2015. "Very deep convolutional networks for large-scale image recognition." *Proc. of International Conference on Learning Representations.* San Diego, USA.

Srivastava N., Hinton G., Krizhevski A., Sutskever A., and Salakhutdinov R. 2014. "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research* 1929-1958.

Szegedy C., Liu W., Jia Y., Sermanet P., Reed S., Anguelov D., Erhan D., Vanhoucke A., and Rabinovich A. 2015. "Going deeper with convolutions." *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition.* Boston, USA: IEEE. 1-9.

Szegedy, C., S. Ioffe, V. Vanhoucke, and A. Alemi. 2016. "Inception-v4, inception-resnet and the impact of residual connections on learning." *Association for Advances in Artifical Intelligence.* San Francisco, USA.

Tuia D., Volpi M., Copa L., Loris M., and Munoz-Mari J. 2011. "A survey of active learning algorithms for supervised remote sensing image classification." *IEEE Journal of Selected Topics in Signal Processing* 606-617.

Viola P., and Jones M. 2001. "Rapid object detection using a boosted cascade of simple features." *Proc. of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition.* Kauai, USA: IEEE. I-I.

Wang, Y. 2014. "An analysis of the Viola-Jones face detection algorithm." *Image Processing On Line* 128-148.

Webb, G. 2000. "Multiboosting: A technique for combining boosting and wagging." *Machine Learning* 159-196.

Wu R., Yan S., Shan Y., Dang Q., and Sun G. 2015. "Deep Image: Scaling up Image Recognition." *arXiv preprint arXiv:1501.02876.*

Zaharia M., Chowdhurry M., Das T., Dave A., Ma J., McCauley M., Franklin M., Shenker S., and Stoica I. 2012. "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing." *Proc. of the 9th USENIX conference on Network Systems Design and Implementation.* Bellevue, USA: USENIX Association. 2-2.