Département de géomatique appliquée

Faculté des lettres et des sciences humaines

Université de Sherbrooke

Classification automatique de nuages de points issus de LiDAR aéroporté par réseau à convolutions continues

Mathieu Turgeon-Pelchat

Mémoire présenté pour l'obtention du grade de Maître ès sciences (M.Sc.),

Cheminement en télédétection

Mai 2021

Directeur de recherche :

Pr. Yacine Bouroubi

Département de Géomatique appliquée, Université de Sherbrooke, Sherbrooke (QC)

Codirecteur de recherche :

Dr. Samuel Foucher

Centre de Recherche en Informatique de Montréal (CRIM), Montréal (QC)

Dr. Nouri Sabo

Centre canadien de cartographie et d'observation de la Terre, Ressources naturelles Canada, Sherbrooke (QC)

Membres du jury :

Pr. Richard Fournier

Département de Géomatique appliquée, Université de Sherbrooke, Sherbrooke (QC)

Membre interne du jury

Dr. Pierre-Luc St-Charles

MILA, Montréal (QC)

Membre externe du jury

Résumé

Les données issues de LiDAR aéroporté permettent de modéliser de façon précise la topographie d'un territoire et sont utilisées dans différents contextes. Le processus de classification du nuage de points permet d'assigner une classe d'occupation du sol à chacun des points. La quantité et la répartition non homogène de ces points complexifient grandement l'automatisation de cette tâche. Pour certaines classes d'objets, les algorithmes traditionnels ne réussissent pas à classifier correctement les points avec un niveau de qualité satisfaisant. C'est le cas pour les classes "bâtiments", "plans d'eau" et "végétation". Ce faisant, cette étape nécessite une intervention manuelle importante afin de corriger la classification effectuée automatiquement, augmentant ainsi le coût de valorisation de ces données. Le succès des algorithmes d'apprentissage profond en vision par ordinateur a mené à une révolution dans différents domaines, notamment en traitement d'images satellitaires. Ces succès ont inspiré diverses recherches sur la classification de nuages de points. La plupart de ces recherches portent sur le traitement de nuages de points issus de LiDAR mobile terrestre, mais elles peuvent très bien être adaptées pour le traitement de données LiDAR aéroporté.

Cette recherche vise à utiliser une méthode d'apprentissage profond pour permettre une automatisation du processus de classification de nuages de points aéroportés, en évaluant une méthode sur deux jeux de données de contextes variés (urbain et rural), puis en modifiant la méthode de sélection des sous-ensembles de points pour qu'elle soit adaptée à la densité locale. La méthode originale a été testée sur deux jeux de données couvrant près de 16 000 km^2, près de Montréal (QC) et de St-Jean (NB) et l'amélioration a été évaluée sur le jeu de données de référence DALES. La méthodologie utilisée est basée sur une adaptation de l'opération de convolution de Boulch (2020) appelée ConvPoint. Cette opération de convolution continue a été utilisée dans une architecture de type encodeur-décodeur qui permet de classifier les points directement, sans avoir recours à une étape de sursegmentation. En expérimentant avec différentes configurations, nous avons obtenu d'excellents résultats d'Intersection-sur-Union (IoU) pour les classes "Végétation moyenne-haute" (93 %) et "Bâtiment" (86 %) sur les ensembles de données de Montréal et Saint-Jean. La taille de bloc adaptative a conduit à certains gains de performances sur le jeu de données de référence DALES ainsi qu'à une sensibilité réduite aux hyperparamètres de la ConvPoint.

Mots-clés : LiDAR, aéroporté, classification automatique, apprentissage profond

Table des matières

Liste des Figures iii

Liste des tableaux v

Liste des équations v

Liste des abréviations vi

Remerciements vii

Avant-Propos viii

1 Introduction 1

1.1 Objectifs 4

1.2 Structure du mémoire 5

2 Revue de littérature 6

2.1 Méthodes qui nécessitent une étape préalable de transformation du nuage de points 6

2.2 Méthodes qui traitent le nuage de point directement 10

3 Classification à grande échelle de nuages de points LiDAR aéroportés par apprentissage profond (Article publié dans le journal canadien de télédétection en mai 2021) 13

3.1 Résumé 13

3.2 Deep Learning-based Classification of Large-scale Airborne LiDAR Point Cloud 13

3.2.1 Abstract 13

3.2.2 Introduction 13

3.2.3 Related work 16

3.2.4 Datasets 21

3.2.5 Experiments 23

3.2.6 Results 26

3.2.7 Discussion 33

3.2.8 Conclusion 34

3.2.9 Acknowledgments 34

4 Adaptation de la taille des blocs en fonction de la densité locale pour la sélection des points dans les nuages de points LiDAR aéroportés à grande échelle pour la méthode de classification ConvPoint (Article soumis au journal canadien de télédétection) 36

4.1 Résumé 36

4.2 Density Dependent Block Size Adaptation for Point Selection in Large-Scale Airborne LiDAR Point Cloud for ConvPoint Classification 36

4.2.1 Abstract 36
4.2.2 Introduction 37
4.2.3 Related work 38
4.2.4 Density Dependent Block Size Adaptation 40
4.2.5 Dataset 42
4.2.6 Experiments 42
4.2.7 Results 44
4.2.8 Discussion 51
4.2.9 Conclusion 52
5 Conclusion 54
5.1 Contribution 54
5.2 Perspectives 55
Références 57
Annexe 1 – Résultats détaillés pour la section 4.2.7. IoU par classe, pour différents “nombre de points”. 61

Liste des Figures

Figure 1.1 — Représentation du nuage de points issus de LiDAR aéroporté avec une vue en plan (image du haut) et en profil (image du bas) d'une zone semi-boisée, située au sud de Montréal. Les variations de couleurs montrent les différences d'élévation. Source des données : Communauté métropolitaine de Montréal, acquisition LiDAR 2018. 2
Figure 3.1 - Schema for ConvPoint's convolutional layer (derived from Boulch, 2020). 20
Figure 3.2 - Maps of the training, validation and test tiles for the Montreal, QC (above) and Saint-Jean, NB (below) datasets. 22
Figure 3.3 - Visual comparisons of the original (left) and corrected (right) classifications for both Montreal and Saint-Jean datasets. 27
Figure 3.4 - Comparison of per class IoUs for XYZ and XYZ with the number of returns and the intensity added. 28
Figure 3.5 - Visual representation of the predicted classes in aerial view (a) with the corresponding imagery (b) and the ground truth (c), along with a 3D view of another area (d) and its ground truth (e). The first half of the Figure comes from the Montreal dataset and the second half is coming from Saint-Jean. 32
Figure 4.1 - Top and side view of a point cloud to show the points distribution and density variability. 41
Figure 4.2 - Histogram showing the variation in point density (pts/m2) for 5 000 randomly selected blocks of 18m in height and width. 45
Figure 4.3 - Average number of points per class (Logarithmic scale) in relation to the local density, for multiple block sizes. 46
Figure 4.4 - Average minimum, maximum and mean elevation (m) in relation to the local density within blocks of 20 m. 47
Figure 4.5 - Comparison of histograms for the average number of points within blocks of fixed size of 20m (orange) and block size adaptation (blue), related to the number of points expected (green). 48
Figure 4.6 - Impact of the block size (m) on the per class IoU, with a number of selected points of 16,336. 49
Figure 4.7 - Mean and per class Intersection over Union (IoU) using the Block Size Adaptation, for 4 configurations of number of points. 50

Figure 4.8 - Visual representation of classification results for our method, the original ConvPoint and the ground truth. 51

Liste des tableaux

Table 2.1 - Comparaison des résultats sur le jeu de données DALES, pour quatre méthodes de classification. Les données sont tirées de Varney *et al.* (2020). 12
Table 3.1 - Comparison of some of the characteristics of the three datasets used in this study. .. 22
Table 3.2 - Hyperparameters tested and their description. 25
Table 3.3 - Summary of the experiments. 26
Table 3.4 - Mean and per class IoU of the original tiles in both datasets. 27
Table 3.5 - Comparison of mean IoUs for different configurations of the number of points and block sizes. 29
Table 3.6 - Mean and per class IoU comparing fine-tuned and trained-from-scratch models on both datasets. 29
Table 3.7 - Mean and per class IoU comparing different dataset configurations for training, tested on the Montreal and Saint-Jean test datasets. 30
Table 3.8 - Comparison of multiple times and IoUs values for training using different numbers of iterations per epoch and inference for different step size values. 31
Table 4.1 - Characteristics of the DALES dataset (derived from Varney *et al.*, 2020). 42
Table 4.2 - Mean Intersection over Union (IoU) for different combinations of "Block Size" and "Number of Points". 48
Table 4.3 - Mean IoU and per class IoU for multiple experiments, comparing our method with the original ConvPoint along with results obtained in Varney *et al.*, (2020). 51

Liste des équations

Equation 3.1 23
Equation 4.1 41
Equation 4.2 44

Liste des abréviations

AP	Apprentissage Profond
CCCOT	Centre canadien de cartographie et d'observation de la Terre (*CCMEO : Canada Center for Mapping and Earth Observation*)
CMM	Communauté Métropolitaine de Montréal
DALES	*Dayton Annotated LiDAR Earth Scan*
DAO	Dessin assisté par ordinateur (*CAD: Computer-Aided Design*)
FCN	*Fully Convolutional Network*
FN	*False Negative*
FP	*False Positive*
GPU	*Graphical Processing Unit*
IoU	*Intersection-Over-Union*
K-NN	*K-Nearest Neighbours*
LiDAR	*Light Detection and Ranging*
MFFP	Ministère des Forêts, de la Faune et des Parcs
MLP	*Multi-Layer Perceptron*
MNE	Modèle Numérique d'Élévation (*DEM : Digital Elevation Model*)
PIR	Proche Infrarouge (*NIR: Near-Infrared*)
RNCan	Ressources naturelles Canada (*NRCan : Natural Resources Canada*)
RVB	Rouge-Vert-Bleu (*RGB: Red-Green-Blue*)
SP	Sécurité Publique (*PS : Public Safety*)
SPG	*SuperPoint Graph*
S3DIS	*Stanford 3D Indoor Scene Dataset*
TP	*True Positive*

Remerciements

Je remercie Yacine Bouroubi, Professeur au Département de Géomatique appliquée de l'Université de Sherbrooke, pour ses conseils judicieux, sa confiance et ses encouragements, qui m'ont permis de mener à bien ce projet.

Merci également à Samuel Foucher, Chercheur au Centre de Recherche en Informatique de Montréal, pour son intérêt, ses réflexions et son temps que je sais précieux. Ses conseils toujours contribués à améliorer la qualité des travaux réalisés dans le cadre de cette recherche.

Je remercie aussi Nouri Sabo, directeur-adjoint du Centre de cartographie et d'observation de la Terre qui m'a offert tout le soutien nécessaire pour réaliser ces travaux.

Je tiens également à remercier la Communauté Métropolitaine de Montréal pour l'accès aux données.

Plus personnellement, merci à ma famille et amis pour leurs encouragements et support. La conciliation travail-étude aurait été beaucoup plus difficile sans vous.

Finalement un merci spécial à ma conjointe (et responsable de l'esthétisme des schémas) Catherine pour son écoute, son aide et sa patience. Je n'aurais pas pu mener ce projet à terme sans toi.

Avant-Propos

Le présent document prend la forme d'un mémoire par articles, composé de cinq chapitres. Les articles (Chapitres 3 et 4) sont présentés en anglais, tel que publié. Le Chapitre 1 introduit la problématique et les objectifs de recherche.

Le Chapitre 2 présente une revue de littérature portant sur la classification LiDAR.

Le Chapitre 3 est composé d'un article sur la classification LiDAR à grande échelle, publié au *Journal canadien de télédétection* : Turgeon-Pelchat, M., Foucher, S., & Bouroubi, Y. (2021). Deep Learning-based Classification of Large-scale Airborne LiDAR Point Cloud. *Canadian Journal of Remote Sensing*. https://doi.org/10.1080/07038992.2021.1927687

Le Chapitre 4 présente une amélioration à la méthode utilisée dans le cadre de cette recherche. Cet article a été soumis au Journal canadien de télédétection et est en attente d'une décision. Turgeon-Pelchat, M., Foucher, S., & Bouroubi, Y. (en attente de publication). Density Dependent Block Size Adaptation for Point Selection in Large-Scale Airborne LiDAR Point Cloud for ConvPoint Classification.

Finalement, le Chapitre 5 présente une conclusion générale, présentant les contributions de cette recherche et les perspectives futures.

1 Introduction

La technologie LiDAR (*Light Detection and Ranging*) mesure le temps d'aller-retour, donc la distance parcourue par une impulsion laser émise par un capteur et réfléchie vers ce dernier par les objets sur le terrain. En connaissant la position exacte de la source d'émission et l'angle de visée, il est possible de connaître la position (x, y, z) exacte des éléments au sol qui ont réfléchi le rayonnement (Nayegandhi et Nimetz, 2018). À chaque cible rencontrée, une partie de l'impulsion est retournée vers le capteur et enregistrée par ce dernier. Une portion du signal est absorbée par la cible et une autre portion est transmise. L'impulsion transmise qui rencontre une autre cible est elle aussi retournée, absorbée et transmise, ainsi de suite, jusqu'à ce que le signal soit complètement absorbé, ou que la puissance soit trop faible pour pouvoir être mesurée (Nayegandhi et Nimetz, 2018). Pour une impulsion donnée, chacun des signaux enregistrés se nomme *retour* et pour chacun des retours le capteur enregistre la position (x, y, z) et l'intensité du rayonnement retourné. Dans certains cas, le capteur LiDAR est couplé à une caméra optique, ce qui permet d'associer des valeurs Rouge-Vert-Bleu (RVB) à la surface visée. La plupart des capteurs LiDAR permettent de mesurer jusqu'à 5 retours, pour une impulsion donnée (Nayegandhi et Nimetz, 2018). La fréquence d'émission des impulsions des capteurs LiDAR récents varie entre 50 000 et 200 000 impulsions à la seconde (Nayegandhi et Nimetz, 2018). L'ensemble des points enregistrés par le capteur est appelé *nuage de points*. Dans un nuage de points suffisamment dense, il est possible d'obtenir plusieurs points au niveau du sol en milieu forestier. La figure 1.1 montre l'organisation et la distribution des points sur une vue en plan et en coupe d'un nuage de points. Le LiDAR aéroporté permet des acquisitions sur de plus grandes zones, réduisant ainsi le coût d'acquisition des données. Cependant, la densité des points du nuage résultant sera grandement diminuée, par rapport à une acquisition terrestre (Maune, 2018).

Les données LiDAR permettent de représenter avec une grande précision la topographie et les formes du terrain. Par exemple, ces données sont utilisées dans le domaine de la foresterie pour le calcul de biomasse et la planification des accès forestiers ainsi que dans le domaine municipal afin de faciliter la planification du développement urbain. Elles sont aussi un intrant important à la mise en place du concept de ville intelligente et servent de données de base pour dériver une cartographie de zones inondables précise (RNCan et SP Canada, 2018; Maune, 2018; Zhao *et al.*, 2018).

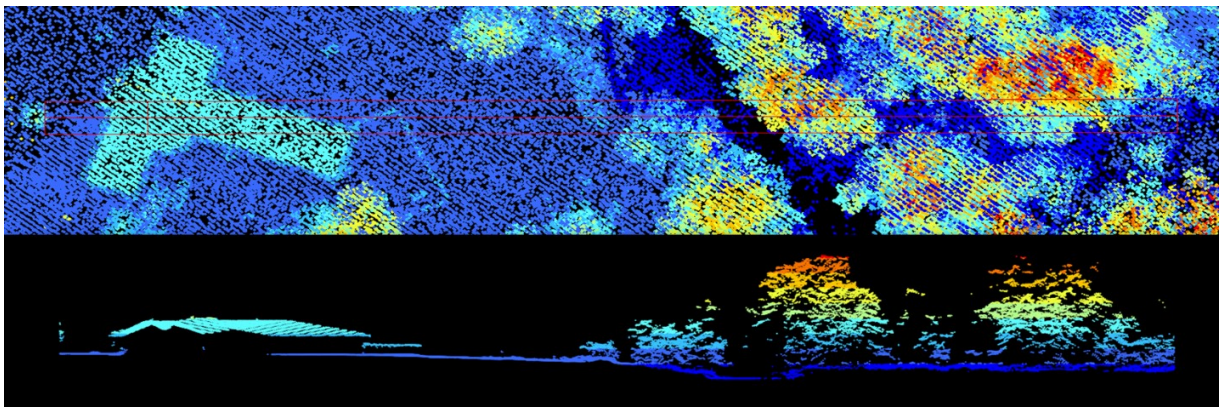


Figure 1.1 — Représentation du nuage de points issus de LiDAR aéroporté avec une vue en plan (image du haut) et en profil (image du bas) d'une zone semi-boisée, située au sud de Montréal. Les variations de couleurs montrent les différences d'élévation. Source des données : Communauté métropolitaine de Montréal, acquisition LiDAR 2018.

Depuis sa commercialisation, dans les années 70, la technologie LiDAR est en évolution rapide. Les avancées technologiques ont amélioré la précision, la couverture et la densité des points acquis par les capteurs. Ces améliorations permettent, entre autres, d'embarquer ces capteurs sur des drones pour une couverture localisée ainsi que sur des plateformes aéroportées pour une acquisition à plus haute altitude, permettant de couvrir une plus grande région. De ces améliorations ont découlé des réductions importantes des coûts d'acquisition des données LiDAR. La valeur importante des données LiDAR et la réduction des coûts d'acquisition mènent actuellement à une augmentation de la quantité de ces données et de leur potentiel d'application.

La plupart des provinces canadiennes ont une stratégie d'acquisition des données LiDAR aéroporté. Par exemple, le Nouveau-Brunswick est entièrement couvert en données LiDAR depuis 2020 et le Québec prévoit terminer la portion sud de la province d'ici la fin 2021 (MFFP, 2021). Au niveau fédéral, le Centre canadien de cartographie et d'observation de la Terre (CCCOT) de Ressources naturelles Canada (RNCan) vise à couvrir le pays en données d'élévation à haute résolution spatiale, dont la partie sud du pays en données LiDAR provenant d'acquisitions réalisées en collaboration avec les provinces et territoires (RNCan, 2017).

Les nuages de points sont majoritairement utilisés afin d'en dériver des modèles numériques d'élévation (MNE) précis (Yousefhussien *et al.*, 2018; Maune, 2018). L'assignation des points à

une classe (d'objet ou d'occupation du sol), aussi appelée classification du nuage de points, permet d'en tirer davantage d'informations, telles que la nature des éléments au sol. La classification des points est requise pour améliorer la qualité des MNE générés, ainsi que pour faciliter l'analyse et l'extraction des éléments au sol. Par exemple, en connaissant les points appartenant aux bâtiments, il est possible de déterminer la pente des toits et ainsi estimer le potentiel photovoltaïque des toits de bâtiments, pour une région donnée.

La structure et la non-uniformité des nuages de points expliquent la difficulté d'en automatiser la classification (Novac, 2018; Yousefhussien *et al.*, 2018; Zhao *et al.*, 2018, Bello *et al.*, 2020). Ces caractéristiques varient énormément d'un projet à l'autre, selon les spécifications d'acquisition, le capteur utilisé et les conditions de la zone au moment de l'acquisition. Pour certaines classes d'intérêts, comme les bâtiments et les plans d'eau, les algorithmes de classification automatique ne permettent pas d'obtenir des résultats se rapprochant de ceux d'un technicien qualifié (Novac, 2018). Ce faisant, l'étape de classification nécessite une intervention manuelle importante afin de corriger les résultats des processus automatiques, ce qui augmente le temps et la difficulté de préparation des données LiDAR et par conséquent, le coût de valorisation de ces données. Une meilleure automatisation de cette tâche, avec des algorithmes rapides et efficaces, permettrait de réduire de manière significative le coût d'exploitation des données LiDAR (Novac, 2018).

Les capacités offertes par les différents capteurs LiDAR aéroportés induisent aussi une variabilité dans les nuages de points. Par exemple, l'intensité de l'impulsion envoyée et reçue dépend des caractéristiques techniques du capteur utilisé et des propriétés de la cible (Jutzi et Gross, 2009; Nayegandhi et Nimetz, 2018; Novac, 2018). Ainsi, les valeurs d'intensité d'un nuage à l'autre présenteront des différences en fonction des capteurs utilisés. Une autre partie de la variabilité des nuages de points provient des structures géométriques complexes des objets au sol (Yousefhussien, 2018). Les impulsions reçues et enregistrées ne sont pas distribuées de manière régulière sur les objets au sol. Les occlusions (forêts, bâtiments, ponts, etc.) vont empêcher l'impulsion laser d'atteindre le sol ou de retourner au capteur et ainsi rendre irrégulière la distribution des points dans le nuage. Le signal laser peut aussi être absorbé par la cible au sol, comme c'est le cas dans l'eau. La longueur d'onde de l'impulsion laser utilisée pour les applications terrestres est située dans le proche infrarouge (PIR), causant une absorption du signal dans l'eau (Nayegandhi et Nimetz, 2018).

Le succès des algorithmes d'apprentissage profond (AP) dans le domaine de la vision par ordinateur a inspiré plusieurs recherches portant sur la classification des nuages de points (Huang et You, 2016; Guo *et al.*, 2019; Xie *et al.*, 2020). La majorité de ces recherches portent sur les nuages de points issus de capteurs LiDAR mobiles terrestres (Qi *et al.* 2017a; Boulch *et al.*, 2018; Landrieu et Simonsovsky, 2018) ou sur ceux issus de données de dessins assistés par ordinateur (DAO) (Wu *et al.*, 2015; Qi *et al.*, 2017a; Qi *et al.*, 2017b). Varney *et al.* (2020) ont récemment présenté un jeu de données de LiDAR aéroporté comportant 10 tuiles de 1 km^2 et 505 millions de points. Ce jeu de données comporte une densité moyenne de 50 points au mètre carré. La classification de ce jeu de données a été réalisée manuellement et validée par plusieurs experts. Par rapport au jeu de données DALES (Varney *et al.*, 2020), les acquisitions LiDAR aéroportés traditionnelles ont une densité de points beaucoup plus faible. En effet, le guide d'orientation des acquisitions LiDAR aéroportées, développé par RNCan, suggère des densités nominales entre 1 et 12 impulsions/m^2, dépendamment de l'application visée (RNCan, 2018). La robustesse de ces algorithmes sur des jeux de données volumineux, face aux variabilités présentes dans les nuages de points et dans des contextes urbains et ruraux demeure à faire afin d'en faciliter l'utilisation.

1.1 Objectifs

L'objectif principal de cette recherche est de faciliter l'utilisation des algorithmes d'AP pour faire la classification des nuages de points issus du LiDAR aéroporté en contexte urbain et rural.

Les objectifs spécifiques portent sur les trois éléments suivants :

- Évaluer les performances d'un algorithme d'AP (architecture ConvPoint) en termes de temps de traitement et de précision de classification des nuages de points;
- Évaluer l'impact sur la qualité de classification de l'utilisation d'information complémentaire sur les points, tels que l'intensité et le numéro du retour;
- Améliorer les performances de la méthode originale pour réduire la sensibilité à certains paramètres liés à l'étape d'entraînement (hyperparamètres) des modèles d'AP et augmenter la qualité de la classification.

Les classes d'intérêts utilisées pour cette recherche visent à classifier le sol, les bâtiments, l'eau et la végétation moyenne-haute. Ces classes sont utilisées dans la plupart des applications du LiDAR aéroporté (RNCan, 2018).

1.2 Structure du mémoire

Ce mémoire contient deux articles qui répondent aux objectifs présentés précédemment. Le chapitre 2 présente une revue de littérature sur les méthodes de classification des nuages de points. Cette revue permet de comprendre les limites des méthodes traditionnelles et justifie le choix de l'algorithme utilisé dans les chapitres subséquents.

Le chapitre 3 contient l'article portant l'évaluation de la méthode ConvPoint (Boulch, 2020) à grande échelle, pour des données de LiDAR aéroporté. Cet article propose une comparaison quantitative et qualitative des performances de cet algorithme sur des jeux de données acquises à Montréal (QC) et à Saint-Jean (NB). L'analyse compare la qualité de classification de ConvPoint selon différentes configurations d'entraînement. Il s'agit, à notre connaissance, de la première fois que ces algorithmes sont utilisés sur une quantité de données de LiDAR aéroporté aussi importante. Les résultats obtenus permettent de mieux comprendre les forces et les limites des algorithmes d'AP. Cet article est publié dans le numéro spécial du *Journal canadien de télédétection – L'apprentissage automatique à grande échelle pour la cartographie au moyen de données multi-sources :* Turgeon-Pelchat, M., Foucher, S., & Bouroubi, Y. (2021). Deep Learning-based Classification of Large-scale Airborne LiDAR Point Cloud. *Canadian Journal of Remote Sensing*. https://doi.org/10.1080/07038992.2021.1927687

Le chapitre 4 présente l'article qui propose une amélioration à la méthode ConvPoint. Cette amélioration permet de réduire la sensibilité à deux hyperparamètres liés à la sélection des points qui sont passés au modèle pour en faire la classification. Ces deux hyperparamètres sont : la taille de la zone (bloc) de recherche et le nombre de points à échantillonner dans cette zone. Cette amélioration permet de simplifier l'utilisation de cet algorithme dans différents contextes, en facilitant la tâche d'entraînement du modèle. Cette amélioration a été testée et validée avec le jeu de données de référence DALES (Varney *et al.*, 2020). Cet article a été soumis au *Journal canadien de télédétection* et est en cours d'évaluation.

Finalement, le chapitre 5 présente une discussion et une conclusion sur les perspectives futures, les pistes d'améliorations, vers une utilisation à grande échelle des algorithmes d'AP pour faire la classification des nuages de points.

2 Revue de littérature

La présente section détaille les méthodes utilisées pour faire la classification des nuages de points et contient deux sous-sections, soient : les méthodes qui nécessitent une étape de transformation du nuage de points, et celles qui traitent du nuage de points directement, sans cette étape préalable. La première sous-section aborde les méthodes traditionnelles (sans utilisation de l'apprentissage automatique), alors que la deuxième recense uniquement des méthodes basées sur l'AP.

2.1 Méthodes qui nécessitent une étape préalable de transformation du nuage de points

Certaines méthodes permettant de faire la classification des nuages de points sont fortement inspirées de la classification orientée-objet en traitement d'image. Ces méthodes comportent les deux mêmes étapes soient : (1) la segmentation du nuage de points et (2) la classification des éléments segmentés (Grilli *et al.*, 2017).

La segmentation vise à regrouper les points ayant des caractéristiques géométriques similaires (Grilli *et al.*, 2017; Xie *et al.*, 2020). Les revues de Grilli *et al.* (2017) et Xie *et al.* (2020) recensent trois groupes de méthodes de segmentation des nuages de points qui n'utilisent pas l'apprentissage automatique. Ces catégories sont les méthodes basées sur : (1) la détection de limites (*edge-based*), (2) la croissance de région (*region growing*) et (3) l'appariement de modèle (*model fitting*) (Grilli *et al.*, 2017; Xie *et al.*, 2020). Les algorithmes *edge-based* comportent deux étapes, soient l'extraction des limites et le regroupement des points entre les limites. L'utilisation de cet algorithme est restreinte en raison de la non-uniformité des nuages de points, qui créent des limites non reliées (Xie *et al.*, 2020). Les algorithmes par croissance de région sont des processus itératifs qui regroupent les points de plusieurs régions ensemble, sur la base de critères qui doivent être configurés par l'utilisateur. Ces algorithmes sont très sensibles aux paramètres ainsi qu'à la sélection des points initiaux, qui serviront de point de départ des régions (Grilli *et al.*, 2017). Les algorithmes d'appariement de modèles utilisent de l'information *a priori* sur les formes géométriques recherchées afin de les détecter dans le nuage de point. Ces algorithmes sont efficaces pour les éléments géométriques simples (Grilli *et al.*, 2017).

Suite à la segmentation du nuage de points, l'étape de classification requiert de construire les descripteurs permettant de caractériser la forme des objets segmentés dans le nuage de points

(Dorninger et Pfeifer, 2008; Daniel, 2018). Daniel (2018) présente une revue des descripteurs 3D pour la reconnaissance d'objets dans les nuages de points acquis par capteurs LiDAR mobiles. Les descripteurs permettent de décrire un objet localement (descripteurs locaux) et globalement (descripteurs globaux). L'utilisation et la combinaison de plusieurs de ces descripteurs permettent de détecter et de classifier certains objets dans le nuage de points (Daniel, 2018). Le développement de ces descripteurs nécessite des connaissances *a priori* sur les géométries des objets à extraire. Par conséquent, plus la diversité des objets augmente, plus les caractéristiques doivent être complexes (Özdemir et Remondino, 2019). Ces éléments limitent l'utilisation de descripteurs génériques, complexifient le développement de ces caractéristiques et réduisent du même coup la possibilité d'utiliser ces algorithmes sur des zones de taille importante (Özdemir et Remondino, 2019). De plus, le développement de caractéristiques doit être fait pour chacune des classes d'information à extraire. Ce processus est donc long, difficile et limité dans son utilisation (Xie *et al.*, 2020; Guo *et al.*, 2019).

Les approches algorithmiques d'AP ont permis de réaliser de grandes percées dans différents domaines tels que la vision par ordinateur, le traitement du langage naturel et plus récemment, l'observation de la Terre (Long *et al.*, 2015; Kampffmeyer *et al.*, 2016; Ball *et al.*, 2017 Scott *et al.*, 2017). Ces succès sont majoritairement redevables à l'amélioration des capacités de calcul, avec l'avènement des *graphical processing units* (GPUs), à de meilleurs algorithmes d'entraînement et à la quantité grandissante de données pouvant être utilisées pour entraîner ces algorithmes (Ronneberger *et al.*, 2015; Goodfellow *et al.*, 2017). En effet, les approches d'AP sont spécialement conçues pour tirer avantage de la quantité phénoménale de données disponibles depuis les dernières années. Ces algorithmes « apprennent » à représenter les données à partir d'une multitude d'exemples. Pour qu'un algorithme d'AP soit performant, il doit être entraîné à partir d'une grande quantité de données afin d'en capter les bonnes caractéristiques. D'une part, la quantité des caractéristiques apprises par un modèle est définie par l'architecture du modèle d'AP. D'autre part, la qualité des caractéristiques apprises dépend surtout du nombre et de la variété des données d'apprentissage. En augmentant la quantité d'exemples, on augmente généralement la variété de ceux-ci, ce qui permet aux modèles d'apprendre des caractéristiques qui sont génériques et représentatives de l'ensemble des possibilités pour la tâche à accomplir (Ball *et al.*, 2017; Goodfellow *et al.*, 2017).

La classification de nuages de points profite de l'engouement pour les algorithmes d'AP et plusieurs revues de littérature s'y consacrent (Guo *et al.*, 2019; Bello *et al.*, 2020; Xie *et al.*, 2020). Le nombre de points pouvant être ingérés par les réseaux de neurones profonds spécialisés dans le traitement de nuage de points 3D est souvent limité puisque les caractéristiques sont calculées pour chacun des points du sous-ensemble, ce qui alourdit énormément le processus (Landrieu et Simonovsky, 2018). C'est pourquoi certaines approches basées sur l'AP utilisent une étape préalable de segmentation des nuages de points.

Les premiers essais en classification de nuages de points issus du LiDAR à partir de méthodes d'AP ont largement été inspirés par les succès des réseaux pleinement convolutifs (*fully convolutional networks* [FCN]) de type encodeur-décodeur à faire de la segmentation d'images. Les FCNs utilisent des convolutions 2D ou 3D sur des matrices dérivées du LiDAR. Ces matrices représentent certaines caractéristiques du nuage de points, afin d'en capter l'information essentielle. Parmi ces matrices, on retrouve les cartes de hauteur, les cartes d'orientations et les cartes de pentes et servent d'intrants dans un FCN, pour classifier les éléments au sol (Hamraz *et al.*, 2018; Zhao *et al.*, 2018). D'autres auteurs ont représenté les nuages de points sous forme de matrices 3D, appelés *voxels* (Wu *et al.*, 2015; Huang et You, 2016; Riegler *et al.*, 2017; Tatarchenko *et al.*, 2017; Wang *et al.*, 2018a). La valeur de chacun des voxels représente l'agrégation des points contenus dans l'espace 3D. Les voxels denses (réguliers) ont été utilisés par Huang et You (2016) pour faire la segmentation de nuages de points issus du LiDAR, combiné aéroporté et mobile. La classification des voxels a été réalisée à l'aide d'un réseau à convolution 3D. Étant donné l'espace mémoire requis par les voxels, la taille du réseau était limitée à 2 couches de convolutions. Les nuages de points issus du LiDAR sont généralement épars et la répartition des points dans l'espace est rarement régulière. Cela implique un encodage inutile et très lourd de voxels qui sont majoritairement vides (Guo *et al.*, 2019; Xie *et al.*, 2020). De plus, la mémoire requise pour traiter l'espace voxelisé augmente de manière exponentielle (cubique) avec la diminution de la taille des voxels. Il y a donc un compromis à faire entre la taille des voxels et le temps de calcul/espace mémoire nécessaire. Ces caractéristiques limitent grandement l'utilisation des approches utilisant les voxels denses (Qi *et al.*, 2017a; Özdemir et Remondino, 2019; Hackel *et al.*, 2018).

Pour réduire le nombre de voxels vides, certains auteurs ont eu recours à l'utilisation d'une représentation de type « octree », dont les voxels sont de tailles variables et permettent de réduire considérablement la quantité d'informations à stocker lors de la segmentation (Tatarchenko *et al.*, 2017). Les espaces vides ainsi que les géométries simples sont représentés avec des voxels de plus grande taille alors que les formes plus complexes sont représentées avec des voxels de plus petites tailles afin de préserver les détails de l'objet segmenté. Cette approche a permis de réduire la quantité de mémoire nécessaire pour gérer les voxels; cependant, les résultats de classification obtenus avec cette approche sont très limités (Tatarchenko *et al.*, 2017). Les transformations de nuage de points en matrices mènent invariablement à une perte d'informations sur les points, ce qui affecte en retour la qualité de la segmentation (Boulch *et al.*, 2018; Hackel *et al.*, 2017). De plus, certaines de ces matrices dérivées requièrent beaucoup de temps de traitement, ce qui limite l'utilisation de ces méthodes sur des zones d'intérêts de grande superficie (Hackel *et al.*, 2017; Hamraz *et al.*, 2018).

Landrieu et Simonovsky (2018) ont utilisé un algorithme de voisinage (*k-nearest neighbors*) afin de regrouper les points ayant des caractéristiques géométriques similaires. Ces regroupements sont ensuite représentés sous forme de graphe, appelé S*uperpoint Graph* (SPG), avant d'être classifiés. Cette représentation possède 3 avantages intéressants. Premièrement, il est beaucoup plus simple pour un réseau de reconnaître et de faire la classification des représentations de formes/objets que d'utiliser des voxels ou encore directement les points. Deuxièmement, la représentation en graphe permet de décrire la relation spatiale et géométrique entre les différents objets, caractéristique qui manquait à certaines approches (Landrieu et Simonozsky, 2018). Troisièmement, la quantité de calculs effectués par le réseau est moindre, par rapport aux autres approches, puisque la taille du graphe est déterminée par le nombre d'objets plutôt que par le nombre de voxels ou de points dans le nuage. La classification des *superpoints* est effectuée à l'aide de la méthode *PointNet* (Qi *et al.*, 2017a), décrite à section 2.2. Les auteurs ont testé leur méthode sur le jeu de données acquis par LiDAR mobile terrestre *Semantic 3D*. La méthode a été testée en y ajoutant les valeurs RVB, mais sans les valeurs d'intensité. En comparant avec 4 autres méthodes, les auteurs ont amélioré significativement les résultats, surclassant le précédent état de l'art de 9 %. Pour démontrer la polyvalence de leur approche, les auteurs ont aussi testé et comparé leur méthode sur le jeu de données *Stanford Large-Scale 3D Indoor Spaces* (S3DIS) qui comprend plusieurs nuages de points d'intérieurs de pièces d'un bâtiment. Les résultats démontrent aussi une amélioration de la

classification, sur l'ensemble des métriques et pratiquement toutes les classes. Cette méthode vient corriger certaines lacunes des approches précédentes, mais comporte cependant deux limitations importantes, qui proviennent de l'algorithme de partitionnement et rendent cette méthode inutilisable sur des volumes importants de données (Thomas *et al.*, 2019; Boulch, 2020). La comparaison de chacun des points et de ses caractéristiques avec ses voisins est une opération extrêmement fastidieuse et les points qui forment les *superpoints* sont présumés appartenir à la même classe (*superpoints* sémantiquement homogènes). Par conséquent, la qualité des *superpoints* réalisés va nécessairement dicter la qualité de la classification qui en sera faite (Landrieu et Boussaha, 2019).

La segmentation des nuages de points est une étape généralement longue à calculer, ce qui en limite grandement l'utilisation sur des jeux de données de grande taille (Lin *et al.*, 2018; Landrieu et Boussaha, 2019; Boulch, 2020). De plus, l'évaluation de la qualité des éléments segmentés est difficile et a un impact direct sur la qualité de la classification qui en est faite. L'ensemble de ces éléments réduit l'intérêt d'utiliser les méthodes qui en dépendent.

2.2 Méthodes qui traitent le nuage de point directement

Plusieurs auteurs ont tenté d'appliquer différentes techniques d'AP directement sur le nuage de points, afin de minimiser le temps de calcul et la dégradation de l'information et d'améliorer les résultats de classification (Guo *et al.*, 2019; Xie *et al.*, 2020). Les revues de Xie *et al.* (2020) et de Guo *et al.* (2019) font état de plus de 50 de ces méthodes uniquement basées sur l'AP, parues depuis 2017. La méthode de Qi *et al.* (2017a) est la première stratégie qui utilise efficacement les réseaux de neurones pour faire la classification directement sur le nuage de points (Landrieu et Boussaha, 2019; Boulch, 2020; Xie *et al.*, 2020). L'architecture développée, appelée *PointNet*, utilise un sous-ensemble de points en entrée et applique des encodages sur ces points. La sélection du sous-ensemble est effectuée de manière aléatoire, dans un rayon donné. Ces encodages sont générés à l'aide d'une cascade de *multi-layer perceptron* (MLP) et permettent au réseau de capturer (encoder) des caractéristiques sur les points du sous-ensemble. Les résultats de ces encodages sont concaténés pour former des caractéristiques globales (pour le sous-ensemble de points) et locales (par point), ce qui permet de classifier chacun des points du sous-ensemble. Les avantages majeurs de *PointNet* sont son efficacité et sa simplicité. *PointNet* offre cependant des performances limitées par son incapacité à tenir compte des relations spatiales entre les points (Qi *et al.*, 2017b; Thomas

et al., 2019; Boulch, 2020). Une version améliorée de la solution a été proposée par Qi *et al.* (2017b) avec *PointNet++* qui corrige en partie les lacunes de la première version. Cette seconde version complexifie grandement l'architecture, augmentant du même coup les temps de traitement (Qi *et al.*, 2017b).

Les convolutions effectuées sur des matrices denses 2D ou 3D sont dites discrètes, c'est-à-dire que l'opération de convolution est effectuée sur une structure en grille (Boulch, 2020). Ces opérations ne peuvent être appliquées de manière adéquate sur le nuage de points, sans modifier la distribution et la position des points du nuage (Thomas *et al.*, 2019; Boulch, 2020). Deux approches qui partagent certains concepts ont adapté avec succès l'opération de convolution au nuage de points. Les deux approches proposent des noyaux (*kernel*) de convolution sont représentés par des points qui varient dans l'espace (Thomas *et al.*, 2019; Boulch, 2020).

La première, *Kernel Point Convolution* (KPConv) (Thomas *et al.*, 2019) traite l'information spatiale (x, y, z) et l'information radiométrique (intensité du retour, par exemple) de la même façon. Ces informations sont passées aux noyaux de convolution. Pour chaque point en entrée, le résultat de chacun des noyaux est concaténé pour donner la réponse de la convolution. Puisque les positions des points en entrée et des points du noyau ne sont pas alignées, la corrélation des points est réalisée en fonction de leur distance relative (Thomas *et al.*, 2019).

Pour l'opération de convolution de ConvPoint, les informations spatiales et radiométriques sont traitées séparément, par deux portion du noyau de convolution distinctes. L'alignement entre la portion spatiale du noyau de convolution et l'information spatiale des points d'entrée est déterminée par un MLP avant d'être concaténée avec l'information radiométrique (Boulch, 2020). L'utilisation du MLP permet au réseau d'apprendre la fonction qui intègre l'information spatiale et rend le modèle invariant aux permutations de l'ordre des points (Boulch, 2020).

Dans les deux approches, la convolution est appliquée sur l'ensemble des points d'entrée, contrairement à la convolution discrète qui est appliquée sur un sous-ensemble de la taille du noyau (Thomas *et al.*, 2019; Boulch, 2020). Dans ce contexte, il est nécessaire d'échantillonner un sous-ensemble de points qui seront injectés dans le réseau. Pour KPConv, la sélection de points est réalisée en fonction d'un rayon de recherche. De cette façon, le nombre de points en entrée est variable, ce qui complexifie le traitement de plusieurs sous-ensembles en parallèle (Boulch, 2020).

Pour ConvPoint, un nombre fixe de points est échantillonné. De cette façon, le traitement en parallèle est facilité puisque tous les sous-ensembles ont le même nombre de points. Cependant, il est possible que les caractéristiques apprises par le réseau soient reliées à l'échantillonnage plutôt qu'à la géométrie des objets dans le nuage de points (Boulch, 2020). Pour les jeux de données de grande taille, ConvPoint utilise une approche hybride où les points sont sélectionnés aléatoirement dans un rayon de recherche défini. Si le rayon de recherche spécifié ne contient pas le nombre de points requis, un suréchantillonage aléatoire est fait (Boulch, 2020).

Les opérations KPConv et ConvPoint ont été intégrées dans une architecture de type encodeur-décodeur. Entre chaque couche de convolution, une réduction du nombre de points (sous-échantillonnage) est nécessaire pour ajouter de la profondeur et réduire le nombre de paramètres pour la portion encodeur du réseau. L'opération inverse (suréchantillonnage) est réalisée dans la portion décodeur du réseau, pour adéquatement classifier chacun des points du nuage de points. Pour ce faire, ConvPoint utilise un algorithme de voisinage (*k-nn*) pour déterminer le nombre de points en sortie de la convolution (Boulch, 2020).

Pour démontrer l'efficacité des méthodes KPConv et ConvPoint, elles ont été testées sur plusieurs jeux de données de référence, dont le jeu de données LiDAR aéroporté DALES (Varney *et al.*, 2020). Le tableau 1 fait la comparaison de PointNet++ (Qi *et al.*, 2017b), SPG (Landrieu et Simonovsky, 2018), KPConv (Thomas *et al.*, 2019) et ConvPoint (Boulch, 2020) sur le jeu de données DALES. Les données sont issues de Varney *et al.* (2020). Les résultats montrent que KPConv performe mieux que les autres méthodes. Les auteurs n'ont cependant pas répertorié les temps de traitement des méthodes, ce qui manque à l'étude.

Table 2.1 - Comparaison des résultats sur le jeu de données DALES, pour quatre méthodes de classification. Les données sont tirées de Varney et al. (2020).

Nom de la méthode	**IoU moyen**	**Précision globale**
KpConv	**0,978**	**0,811**
ConvPoint	0,972	0,674
PointNet++	0,957	0,683
SPG	0,955	0,606

3 Classification à grande échelle de nuages de points LiDAR aéroportés par apprentissage profond (Article publié dans le journal canadien de télédétection en mai 2021)

3.1 Résumé

Les données LiDAR aéroportées permettent la modélisation précise de la topographie et sont utilisées dans de multiples contextes. Pour en faciliter les analyses sur ces données, le processus de classification des nuages de points permet d'attribuer une classe, d'objet ou d'élément au sol, à chaque point. La présente recherche utilise ConvPoint, une méthode d'apprentissage profond, pour effectuer la classification de nuages de points aéroportés, dans des contextes ruraux et urbains de grandes superficies. Plus précisément, nos expériences sont situées près de Montréal (QC) et de Saint-Jean (NB) et notre approche est conçue pour classifier cinq classes, soient: "Bâtiment", "Sol", "Eau", "Végétation basse" et "Végétation moyenne-haute". En expérimentant avec différentes configurations, nous avons obtenu d'excellents résultats d'Intersection-sur-Union (IoU) pour les classes "Végétation moyenne-haute" (93 %) et "Bâtiment" (86 %) sur les deux ensembles de données et nous fournissons des indications pour améliorer les temps de traitement ainsi que la précision.

3.2 Deep Learning-based Classification of Large-scale Airborne LiDAR Point Cloud

3.2.1 Abstract

Airborne LiDAR data allow the precise modelling of topography and are used in multiple contexts. To facilitate further analysis, the point cloud classification process allows the assignment of a class, object or feature, to each point. This research uses ConvPoint, a deep learning method, to perform airborne point cloud classification at scale, in rural and urban contexts. Specifically, our experiments are located near Montreal (QC) and Saint-Jean (NB) and our approach is designed to classify five classes; we used “Building”, “Ground”, “Water”, “Low Vegetation” and “Mid-High Vegetation”. Experimenting with different configurations, we achieved excellent Intersection-over-Union results for the “Mid-High Vegetation” (93 %) and “Building” (86 %) classes on both datasets and provide insights to improve processing times as well as accuracy.

3.2.2 Introduction

Airborne LiDAR (light detection and ranging) data are largely used in domains such as forestry for biomass calculation and forest access planning, as well as in municipal domains for urban planification and floodplain mapping (NRCan and PS, 2018; Maune, 2018; Zhao *et al.*, 2018). The

evolution of LiDAR technology has led to extensive enhancement of the sensors' acquisition capacities regarding the precision, coverage and point density. Those enhancements have led to reduced acquisition costs for LiDAR data, thereby increasing the amount of data being acquired. For example, the Canada Centre for Mapping and Earth Observation (CCMEO) of Natural Resources Canada (NRCan) intends to cover the country in high-resolution data using LiDAR data for the southern portion (NRCan, 2020). To carry out this strategy, the CCMEO and its partners (provincial and territorial authorities, other federal departments and municipalities) regularly acquire point clouds from airborne LiDAR, the primary objective of which is to facilitate the mapping of flood-prone areas. As of September 2020, 385,000 km^2 in Canada had been covered by LiDAR data (NRCan, 2020).

Point clouds are mainly used to derive Digital Elevation Models (DEMs), but this highly accurate data can also be used for various types of analyses (Yousefhussien *et al.*, 2018; Maune, 2018). To facilitate the analysis of point clouds, each point must be associated with a class of object or feature, in a process called point cloud classification. For some classes, such as buildings and water bodies, automatic classification algorithms do not provide results that come close to what a qualified technician can achieve in terms of accuracy (Novac, 2018). Classifying buildings, water bodies and ground features is crucial to derive floodplain mapping from LiDAR data (NRCan and PS Canada, 2018). The classification step still requires significant manual intervention to correct the results of the automatic processes, which increases the time and the complexity involved in preparing LiDAR data, thereby making this task costly.

The structure and non-uniformity of point clouds contribute to the difficulty of automating the classification task (Novac, 2018; Yousefhussien *et al.*, 2018; Zhao *et al.*, 2018; Bello *et al.*, 2020). These characteristics vary greatly from one project to another, depending on the acquisition specifications, the sensor used and the conditions at the time of acquisition. NRCan has developed an orientation guide for airborne LiDAR acquisitions with a view to standardizing acquisition methods and thereby reducing some of the variability observed between data sets (NRCan and SP Canada, 2018). That guide aims to improve the understanding of point cloud quality, helping both the contract supplier and the contracting party. It also proposes acquisition best practices (NRCan and PS, 2018). For example, the nominal pulse density is a measure that qualifies the number of pulses per unit area for an entire LiDAR project (NRCan and PS, 2018). This value ensures a

quantity of points that meets the requirements of the contract supplier during the planning phase of LiDAR projects. The recommended nominal density for urban applications, noted in the Federal Airborne LiDAR Data Acquisition Guideline, should be a minimum of 10 pulses/m^2 (NRCan and PS, 2018). However, to meet the requirements of the intended application, it may be recommended that the required nominal density be adjusted in the acquisition plan. Improved automatic classification processes designed to function in reduced point density could allow flights at higher altitudes, thereby reducing the acquisition costs of airborne LiDAR (Novac, 2018).

The capabilities offered by the various airborne LiDAR sensors also contribute to increased variability in point clouds. For example, the intensity of the sent and received pulses depends on the technical characteristics of the sensor and the properties of the target (Jutzi and Gross, 2009; Nayegandhi and Nimetz, 2018; Novac, 2018).

Another part of the variability in point clouds comes from the complex geometric structures of ground objects (Yousefhussien *et al.*, 2018). The received and recorded pulses are not evenly distributed over the ground objects. Occlusions (forests, buildings, bridges, etc.) will prevent the laser pulse from reaching the ground or from returning to the sensor and thus make the distribution of points in the cloud uneven. The laser signal can also be absorbed by the target on the ground, as is the case in water and sometimes rooftops. The wavelength of the laser pulse used in land applications is in the near infrared (NIR) range, causing the signal to be absorbed in water (Nayegandhi and Nimetz, 2018).

The success of deep learning algorithms in the field of computer vision has inspired several studies on point cloud classification (Huang and You, 2016; Guo *et al.*, 2019; Xie *et al.*, 2020). This research has focused mainly on land-based mobile LiDAR (Qi *et al.*, 2017a; Boulch *et al.*, 2018; Landrieu and Simonsovsky, 2018) or computer-aided design (CAD) point clouds (Wu *et al.*, 2015; Qi *et al.*, 2017a; Qi *et al.*, 2017b). However, these approaches can be adapted to classify data acquired by airborne LiDAR (Yousefhussien *et al.*, 2018; Varney *et al.*, 2020). In order to promote a better use of these methods, we need to validate the robustness of these approaches to the variabilities present in point clouds and in different contexts.

3.2.2.1 Objectives

This study aims to assess the applicability of a deep learning method for large-scale airborne LiDAR point cloud classification. Specifically, we tested the ability of ConvPoint (Boulch, 2020) to perform the classification of buildings, waterbodies, vegetation and ground points in different contexts (urban and rural) and using multiple network configurations. The performances were assessed both in terms of accuracy and processing time for training and inference in order to develop insights on the use of deep learning for this task.

We first present a summary of the related approaches for point cloud classification, followed by a description of our experimental setups and the ConvPoint (Boulch, 2020) method used. Next, we present our results and discuss our interpretations and lessons learned from our experiments and results. The last section concludes this study and provides insights for improving classification results.

3.2.3 Related work

Point cloud classification methods can be separated into two groups: methods that require a preprocessing segmentation step before classifying the segmented objects (Grilli *et al.*, 2017; Daniel, 2018), and methods that use the point cloud directly, without a preprocessing step. After the preprocessing segmentation step, the classification process in the first group of methods can then be based on either deep learning or traditional methods. All of these methods are fairly recent, particularly for those based on deep learning techniques (Thomas *et al.*, 2019; Boulch, 2020).

3.2.3.1 Point Cloud Segmentation

Point clustering is mainly used to reduce the space required for processing the point cloud (Landrieu and Simonozsky, 2018; Landrieu and Boussaha, 2019; Xie *et al.*, 2020). To achieve point clustering, some authors have opted for an approach in which the points are represented as volumetric pixels, called voxels (Wu *et al.*, 2015; Huang and You, 2016; Riegler *et al.*, 2017; Tatarchenko *et al.*, 2017; Wang *et al.*, 2018). Huang and You (2016) used dense (regular) voxels to segment point clouds from combined airborne and mobile LiDAR data. LiDAR point clouds are generally scattered and the distribution of points in space is rarely regular, which leads to unnecessary and very heavy encoding of empty voxels (Guo *et al.*, 2019; Xie *et al.*, 2020). In addition, the amount of memory required to process voxelized space increases exponentially (cubically) as the voxel size decreases. These characteristics greatly limit the use of dense voxel approaches (Qi *et al.*, 2017a; Özdemir and Remondino, 2019; Hackel *et al.*, 2018). To reduce the

number of empty voxels, Tatarchenko *et al.* (2017) used an "octree" representation, where the voxels are of varying sizes.

Another over-segmentation method represents the segmented point cloud as graphs. Landrieu and Simonovsky (2018) used the Cut-pursuit algorithm (Landrieu and Obozinski, 2017), based on the k-nearest neighbours (k-NN) algorithm, for the point comparison. Their approach utilizes geometric features (linearity, planarity, dispersion and verticality) to guide the over-segmentation of the point cloud. Classification methods that require a prior over-segmentation step are generally time-consuming. On the other hand, the number of elements to classify is greatly reduced from the original point cloud, allowing the use of classification methods that require more computing power (Lin *et al.*, 2018; Landrieu and Boussaha, 2019; Boulch, 2020).

3.2.3.2 Traditional Methods for Cluster Classification

Traditional methods for classifying segmented point clouds involve constructing descriptors to characterize the shape of objects in the scatter plot (Dorninger and Pfeifer, 2008; Daniel, 2018). Daniel (2018) presents a review of 3D descriptors for the recognition of objects in point clouds acquired by mobile LiDAR sensors. The descriptors allow an object to be described locally and globally, allowing the detection and classification of certain objects in the point cloud (Daniel, 2018). The development of these descriptors requires a priori knowledge of the geometries of the objects to be extracted. This limits the use of generic descriptors, making the development of these features more complex and reducing the possibility of using these algorithms on large areas (Özdemir and Remondino, 2019). In addition, a development of characteristics must be done for each class of information to be extracted, making this process long, difficult and limited in its use (Xie *et al.*, 2020; Guo *et al.*, 2019).

3.2.3.3 Deep Learning Techniques for Point Cloud Classification

Deep learning approaches have led to major breakthroughs in areas such as computer vision, natural language processing, and, more recently, Earth observation (Long *et al.*, 2015; Kampffmeyer *et al.*, 2016; Ball *et al.*, 2017; Scott *et al.*, 2017). Point cloud classification is also benefiting from the popularity of deep learning algorithms and is the subject of several reviews (Guo *et al.*, 2019; Bello *et al.*, 2020; Xie *et al.*, 2020).

The first attempts to classify LiDAR point clouds using deep learning methods were largely inspired by the success of fully convolutional networks (FCNs) in image segmentation. Those

methods require transforming the point cloud into 2D matrices or 3D voxels. The derived matrices represent certain characteristics of the point cloud, capturing its essential information (Hamraz *et al.*, 2018; Zhao *et al.*, 2018). The transformation of point clouds invariably leads to a loss of information, which in turn affects the quality of the classification (Boulch *et al.*, 2018; Hackel *et al.*, 2017). In addition, some of these derived matrices require considerable processing time, which limits the use of these methods over large areas (Hackel *et al.*, 2017; Hamraz *et al.*, 2018).

Landrieu and Simonovsky (2018) use a neighbourhood algorithm to group points with similar geometric characteristics. The clusters are then represented in the form of a graph, called the Superpoint Graph (SPG). The graph representation allows the spatial and geometric relationships between the different objects to be described, a characteristic missing in most approaches (Landrieu and Simonozsky, 2018). This method has corrected some of the shortcomings of previous approaches and has significantly improved the results on multiple benchmark datasets (Landrieu and Simonozsky, 2018). However, it has two important limitations, which stem from the partitioning algorithm, making this method unusable on large volumes of data (Thomas *et al.*, 2019; Boulch, 2020). More precisely, these limitations are (1) the comparison of each point and its characteristics with its neighbours is an extremely tedious operation, and (2) the points forming the superpoints are presumed to belong to the same class (semantically homogeneous superpoints). Consequently, the quality of the superpoints produced will necessarily dictate the quality of the classifications that will be made of them (Landrieu and Boussaha, 2019). Evaluating the quality of superpoints is difficult and time-consuming, which is why it would be preferable to use a method that does not require this preprocessing step.

Several authors have attempted to apply different deep learning techniques directly on the point cloud to minimize computation time and information degradation and improve classification results (Guo *et al.*, 2019; Xie *et al.*, 2020). Reviews by Xie *et al.* (2020) and Guo *et al.* (2019) report more than 50 methods based solely on deep learning since 2017. The method of Qi *et al.* (2017a), called PointNet, is the first strategy that effectively uses neural networks to perform classification directly on the point cloud (Landrieu and Boussaha, 2019; Boulch, 2020; Xie *et al.*, 2020). PointNet encodes the global and local characteristics of a subset of points using a cascade of multi-layer perceptron (MLP). The major advantages of PointNet are its efficiency and simplicity. However, PointNet's performance is limited by its inability to account for spatial

relationships between points (Qi *et al.*, 2017b; Thomas *et al.*, 2019; Boulch, 2020). With PointNet++, Qi *et al.* (2017b) proposed an improved version, partially correcting the shortcomings of the first version. This second version greatly increases the complexity of the architecture, significantly increasing the processing time (Qi *et al.*, 2017b; Varney *et al.*, 2020).

Convolutions on dense 2D or 3D matrices (i.e. on regular grid structures) are said to be discrete (Boulch, 2020). These operations cannot be adequately applied to the point cloud without changing the distribution and positions of the points in the cloud (Thomas *et al.*, 2019; Boulch, 2020). Two approaches that share some concepts have successfully adapted the convolution operation to the point cloud. Both approaches propose kernels of convolution represented by points that vary in space (Thomas *et al.*, 2019; Boulch, 2020). The KPConv (Thomas *et al.*, 2019) and ConvPoint (Boulch, 2020) convolutions can be used in encoder-decoder like architectures, similar to those used in image processing. These two methods were compared on several reference datasets, including the DALES airborne LiDAR dataset (Varney *et al.*, 2020). KPConv slightly outperforms ConvPoint in terms of accuracy, while the training of ConvPoint can be parallelized, making it more suitable for very large datasets. Although ConvPoint's classification accuracy is slightly lower than KPConv's, the ConvPoint convolution differs in two interesting ways that are not covered by KPConv. First, ConvPoint uses an MLP to learn the spatial relationship between the position of the convolution kernel and the input points, which should be an improvement on using distance weighting, as in KPConv (Boulch, 2020). The second difference is in the selection of points at each iteration. ConvPoint uses a fixed number of points within a defined neighbourhood, called block while KPConv uses all points within a defined neighbourhood. This difference allows ConvPoint to process several subsets of points in parallel, thereby considerably reducing processing times (Boulch, 2020). For the purpose of this research, we chose to evaluate ConvPoint for these specific reasons. Figure 3.1 offers a schematized view of the convolutional layer, as described in Boulch (2020).

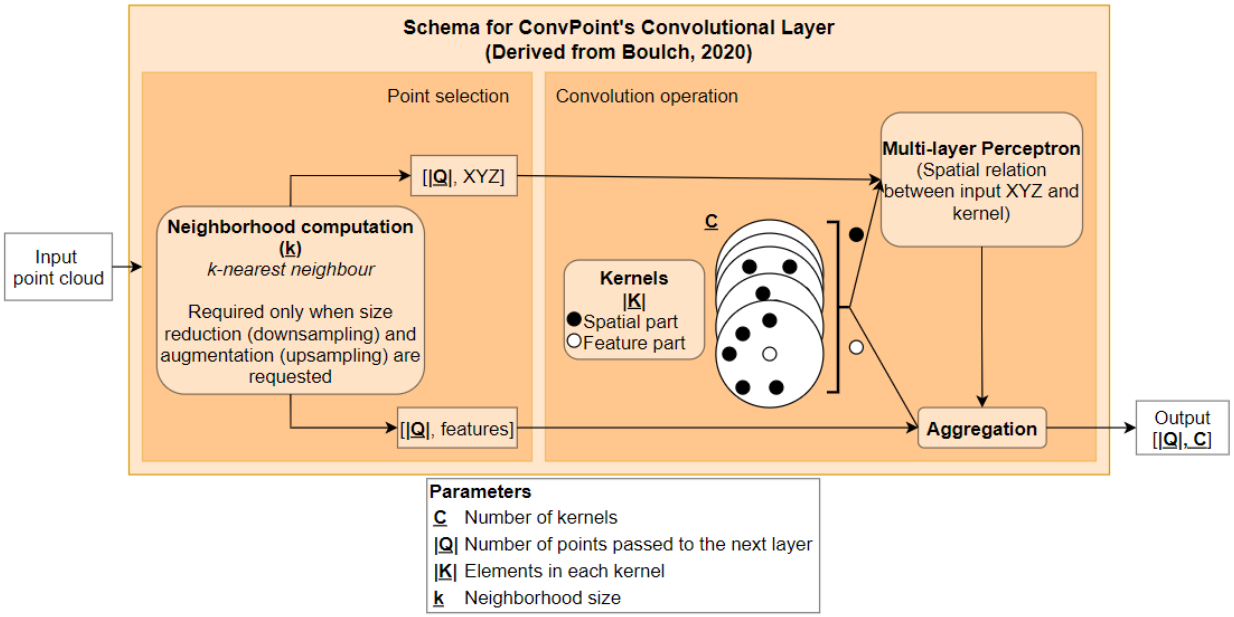


Figure 3.1 - Schema for ConvPoint's convolutional layer (derived from Boulch, 2020).

In the ConvPoint convolution, the spatial and features information are treated separately. The relation between the kernel points and the spatial information of the input points is determined by an MLP before being concatenated with the feature information that has been multiplied by the feature part of the kernels (Boulch, 2020). The use of the MLP allows the network to learn the function that integrates the spatial information and makes the model invariant to permutations in the points' order (Boulch, 2020). This convolution operation is integrated in an encoder-decoder like architecture. In such case, size reduction (downsampling) or augmentation (upsampling) is required to add depth and reduce the number of parameters within the network. In such operations, a k-nearest neighbour algorithm is performed to determine the number of points output from the convolution (Boulch, 2020).

Convolutions are applied to all entry points, unlike discrete convolutions, which are only applied to a subset of the kernel size (Thomas *et al.*, 2019; Boulch, 2020). For large point clouds, it is necessary to sample a subset of points that will be passed to the network. For ConvPoint, a fixed number of points is randomly sampled, within a defined search block, which facilitates parallel processing since all subsets have the same number of points. However, if the number of points to sample and the block size are not set properly, the features learned by the network may be related to sampling rather than to the spatial organization of the points in the point cloud (Boulch, 2020;

Varney *et al.*, 2020). If the specified search radius does not contain the required number of points, random oversampling is performed (Boulch, 2020).

3.2.4 Datasets

Three datasets were used to assess the applicability of deep learning models in rural and urban contexts. The urban dataset is located near Montreal (QC) and the rural one is located around Saint-Jean (NB). The third dataset named DALES (Varney *et al.*, 2020) is used for pre-training, a widespread technique using the features from an already trained model as initialization for another task or dataset (Ball *et al.*, 2017). In our case, we fine-tuned our models on the Montreal and Saint-Jean datasets, using pre-trained features from the DALES dataset.

The Saint-Jean dataset is available under an open license on the province's open data portal. The Montreal data were provided by the Communauté Métropolitaine de Montréal (CMM) and should be available under an open license in the coming year. Both datasets come from airborne LiDAR projects carried out at different dates, using different sensors and with different acquisition specifications. The datasets cover a total area of nearly 16,000 km^2, divided into tiles of 1 km^2. The acquisition and classification were carried out in 2018 on behalf of the CMM and the Province of New Brunswick. Both datasets contain the same five classes, namely "Buildings", "Ground", "Water", "Low Vegetation" and "Mid-High Vegetation", with all remaining points classified as "Other". The DALES dataset is located around the city of Surrey, British Columbia and is comprised of eight classes (Ground, Vegetation, Buildings, Cars, Fences, Power lines, Trucks and Poles). The difference in classes is not of importance as the DALES dataset is only used for pre-training purposes. As shown in Figure 3.2, a subset of tiles from Montreal and Saint-Jean were selected, comprising the training, validation and test datasets. The method for selecting the tiles is described in the experiments section.

For the Montreal region, we have access to a set of aerial images (0.05 m spatial resolution) acquired over the same period as the LiDAR data. The province of New Brunswick also made aerial imagery (0.07 m spatial resolution) available to the public. However, this imagery was acquired in October and November 2015, a few years prior to the LiDAR acquisition. Both of these images were used to evaluate the results. Table 3.1 summarizes some characteristics of the data used for this study.

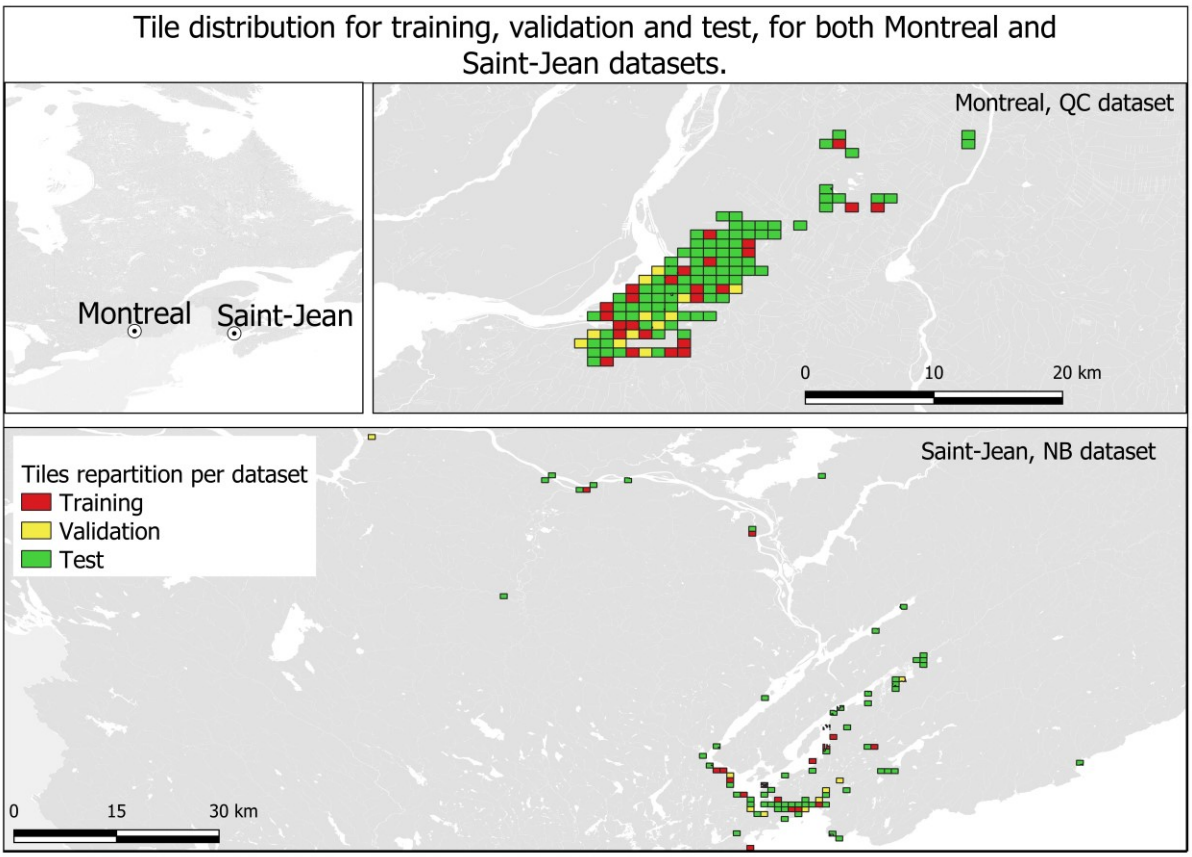


Figure 3.2 - Maps of the training, validation and test tiles for the Montreal, QC (above) and Saint-Jean, NB (below) datasets.

Table 3.1 - Comparison of some of the characteristics of the three datasets used in this study.

Characteristics		**Montréal, QC**	**Saint-Jean, NB**	**DALES (Varney *et al.*, 2020)**
Average nominal density		15 pts/m^2	13 pts/m^2	50 pts/m^2
Minimum nominal density (1 km^2 tile)		4 pts/m^2	4 pts/m^2	unknown
Maximum nominal density (1 km^2 tile)		18 pts/m^2	29 pts/m^2	unknown
Number of points (Millions)		2,439	1,619	505
Number of selected tiles (1 km^2)		123	102	40 (tiles of size 500 m X 500 m)
Point %	Other	70.1	0.6	3.3
	Ground	13.2	33.4	48.9*
	Buildings	12.3	3.7	15.8
	Low Vegetation	0.9	12.8	*fused with ground points
	Mid-High Vegetation	27	41.5	32.0
	Water	4.2	8.0	n/a

3.2.5 Experiments

3.2.5.1 Data Preparation

We selected a subset of tiles from the Montreal and Saint-Jean datasets to create our training, validation and test dataset. Three criteria were used to make this selection. First, we removed the tiles whose points had not been classified. Secondly, we kept tiles with a representation of points in the "Building" class of 10% for Montreal and 1% for Saint-Jean as well as tiles with a representation of "water" points of 5 % for Montreal and 1 % for Saint-Jean. Finally, we kept the tiles with a minimum point density of 4 pts/m2. The tiles were randomly split into training, validation and testing datasets, with approximate percentages of 30, 15 and 55, respectively. Most of the data is used in the test datasets in order to assess the model's ability to be put into production and to perform well on large datasets.

For each dataset, we applied two data preparation steps. The first step transforms the absolute coordinate values of the points (projected coordinates) into values in a relative space (relative coordinates), where the coordinates of the points vary between (0, 0) and (1000, 1000) for each tile. The elevation values are relative to the lowest point of the tile. This transformation ensures that the relationships learned between the input (coordinates) and the output (classes of interest) are not related to a certain projection. The intensity values were normalized between 0 and 1 according to the minimum and maximum values for each tile (Jutzi and Gross, 2009) in the second step.

3.2.5.2 Qualitative and Quantitative Evaluations

The Intersection-over-Union (IoU) value serves as an evaluation metric and was calculated based on the original classification, for the validation and test sets and for each class. Equation 1.1 shows the calculation for the IoU:

$$IoU = \frac{TP}{TP + FP + FN}$$

Equation 3.1

where TP is the true positive, FP is the false positive and FN is the false negative.

The IoU value on the validation set was used during training and allows to quantify the learning performance of the models. This metric was also calculated at the end of the training on the test dataset and serves as the final evaluation of the models. Calculated on the entire set of points, the

IoU value is widely used to evaluate model performance on multi-class point cloud sets (Wu *et al.*, 2015; Armeni *et al.*, 2016; Varney *et al.*, 2020).

Moreover, we performed a visual inspection on part of the test set with the model that obtained the best metrics, and compared the results to the original classification and to the aerial imagery. This visual inspection allows us to qualify the results obtained by our model as well as to have an appreciation of the quality of the original classification.

3.2.5.3 Dataset Quality Assessment

Both Montreal and Saint-Jean datasets have been classified using undisclosed methods, by two different data providers. We conducted a manual correction of the classification for two tiles in each test dataset to assess its general quality. The correction was done using the aerial imagery and 3D view of the point cloud. The amount of time and effort required to manually correct those tiles has limited us to this small subset. The two tiles selected for each dataset were selected to include a good point representation for the "Building" and "Water" classes. Once the correction completed, we compared the original classification with the corrected one using the IoU value to assess its general quality.

3.2.5.4 Point Cloud Classification

This section describes the classification process based on the ConvPoint method (Boulch, 2020). Model training is an empirical process, guided by the work of the authors of the original model. Indeed, the selection of the hyperparameters, which control how the training is performed, depends on the training data set (Goodfellow *et al.*, 2017). Different configurations were tested on a small subset of the Montreal dataset to find an optimal set of hyperparameters. They were then used on both complete datasets to assess the models' performances. A description of each hyperparameters is provided in Table 3.2.

In addition to the spatial information (x, y, z) on the points, it is possible to add other features, such as the intensity and the number of the returns. Different model configurations are tested, with and without the addition of this information in order to assess their impact. Points with or without added information are passed to the ConvPoint segmentation model. As an output, the model produces a probability for each of the classes and each of the points. During training, the predictions are compared to the annotations for loss calculation. The loss calculation is based on cross-entropy, a method well-adapted to multi-class problems (Goodfellow *et al.*, 2017; Boulch,

2020). The updating of the model weights was performed by the Adam optimization algorithm, as defined in Boulch's (2020) method.

Table 3.2 - Hyperparameters tested and their description.

Parameter	**Description**
Characteristics	Characteristics associated with each point: [X, Y, Z] or [X, Y, Z, Number of returns, Intensity]
Block size	Size of the search block from the central randomly-selected point
Number of points	Number of points sampled from each block
Batch size	Number of blocks per iteration
Number of iterations	Number of batches per epoch
Number of epochs	Number of times each iteration is calculated
Learning rate	Step size for the update of the model encodings

The number of points processed for an iteration depends on the size of the batch and the number of points for each of the subsets. The points are loaded in memory on the GPU, and thus their total is dependent upon the capacity of the GPU used. The block size controls the size of the neighbourhood search and the number of points represent the amount of points randomly sampled within each of these blocks. If the amount of points within the block is inferior to the number of points to sample, points are artificially created (oversampled) and their classification is based on the closest neighbour (Boulch, 2020). The two parameters "block size" and "number of points" depend on the point distribution within the point cloud and the configurations tested for those parameters are based on the average density of the point clouds. The distribution of points being variable within the tiles, the size of the blocks and the number of points to be selected must therefore be found empirically. If the number of points to select is too large compared to the size of the search block, there will be too much oversampling and consequently a decrease in performance. Conversely, if the number of points selected is too small compared to the size of the blocks, the effects of the sampling points inside the blocks will be magnified and there will also be a decrease in performance.

A trained model can be used on the test dataset to measure its performance. In each test set, the selection of the starting points is done systematically, with a step of fixed size reducing the impact

of the random selection within the blocks (Boulch, 2020). The predictions for all classified points are aggregated and points that have not been selected are classified according to their closest neighbour.

In addition to the hyperparameters evaluation, we compared the impact of pretraining on models' performance and assessed how a model trained on one dataset can generalize on another. All experiments are summarized in table 3.3.

Table 3.3 - Summary of the experiments.

Experiment	Description	Dataset
Hyperparameter search	Characteristics : [X, Y, Z] or [X, Y, Z, Number of returns, Intensity]	Subset Montreal
	Block size (m) [35; 50; 75; 100]	Subset Montreal
	Number of points [8,168; 16,336; 32,672; 65,344]	Subset Montreal
	Number of iterations [500; 1,000; 5,000]	Complete Montreal
	Number of epochs [10; 50; 100; 200]	Subset Montreal
	Learning rate [0.01; 0.005; 0.001; 0.0005; 0.0001]	Subset Montreal
Impact of pretraining	Pretraining on DALES before fine-tuning or Training from scratch	Complete Saint-Jean and complete Montreal
Generalization	Training using Montreal or Saint-Jean or both datasets	Complete Saint-Jean and complete Montreal
Inference time	Adjusting the step size (m.) at inference [5; 10; 15; 20; 25; 30]	Complete Montreal

3.2.6 Results

3.2.6.1 Dataset Quality Assessment

The classification for two representative tiles for each dataset (Montreal and Saint-Jean) were manually corrected. Table 3.4 shows the resulting IoU when comparing the original classification with the corrected one, for both datasets and Figure 3.3 shows a 3D view on an urban sector for both datasets before and after corrections. One striking element for Montreal is that most "Ground" points were originally classified as "Other" and can be seen both visually and in the IoU values for these classes. There was also some "Other" and "Ground" points in the "Water" areas for Montreal, as depicted in the IoU for the "Water" class. For Saint-Jean, most of the errors were misclassification of "Mid-High vegetation" that should have been classified as "Other", such as

power lines and cars. Overall classification for Saint-Jean was excellent whereas for Montreal, the "Building", "Low vegetation" and "Mid-High Vegetation" classes were generally well classified and corrections were marginal. There are still errors in the point cloud after correction, due to the difficulty in manually classifying the points. Nonetheless, these remaining errors represent less than 5 % of the points.

Table 3.4 - Mean and per class IoU of the original tiles in both datasets.

Dataset	Mean IoU	IoU					
		Other	Building	Water	Ground	Low Vegetation	Mid-high Vegetation
Montreal	0.699	0.142	0.979	0.821	0.296	0.979	0.978
Saint-Jean	0.966	0.842	0.974	0.986	0.996	0.996	0.999

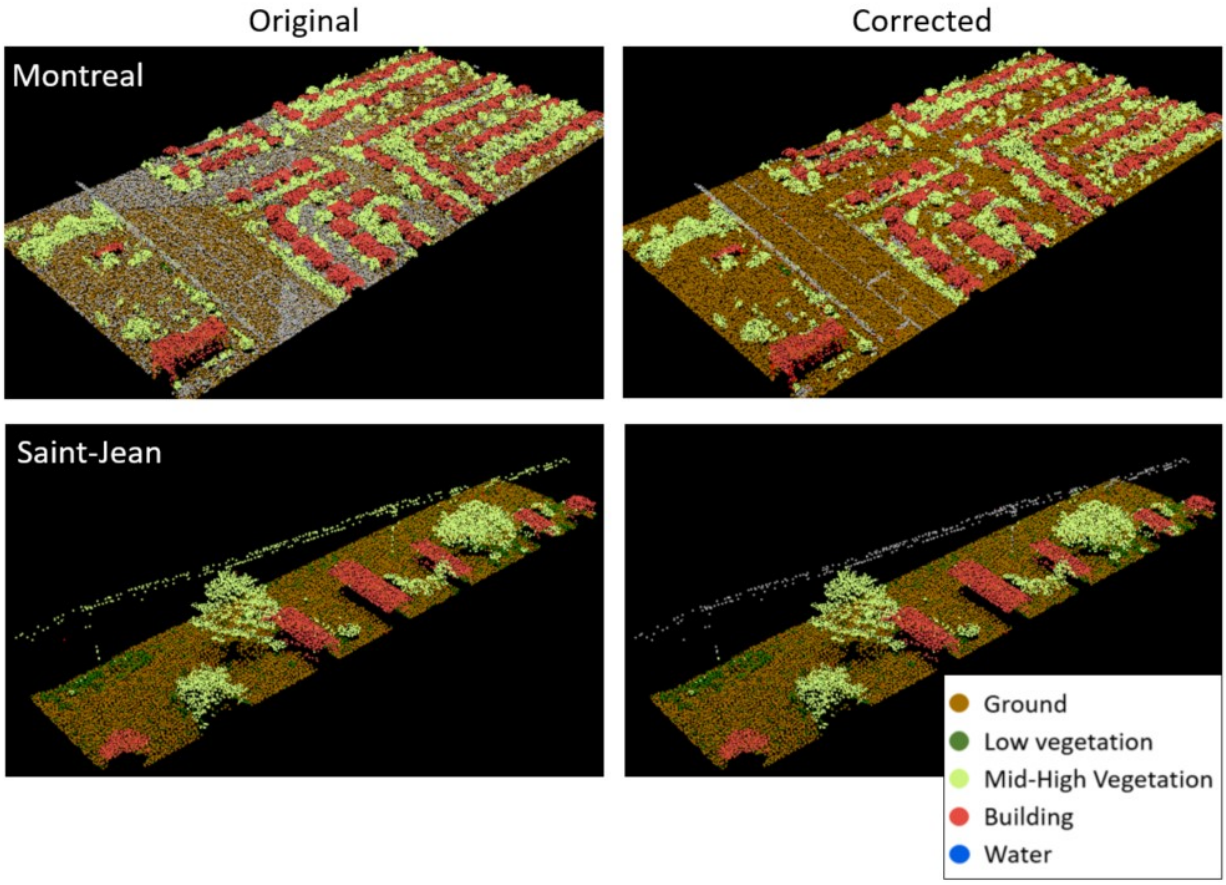


Figure 3.3 - Visual comparisons of the original (left) and corrected (right) classifications for both Montreal and Saint-Jean datasets.

3.2.6.2 *Hyperparameter Search*

We conducted several experiments on a small subset of the Montreal dataset to determine the best set of hyperparameters. This subset was composed of 11 tiles for training and 6 for validation, which allowed for a thorough hyperparameter search.

The first comparison was to verify the effect of using additional information on the points. Figure 3.4 shows the IoU for each class of interest (Buildings, Water, Ground, Low Vegetation and Mid-high Vegetation) with and without incorporating the intensity and the number of returns as additional information on the point coordinates. As indicated in Figure 3.4, almost no differences in performance were observed with the use of additional information on the intensity and the number of returns on the points. When performing manual classification, those features are usually of particular interest to distinguish water from ground and vegetation from buildings and ground. It should be noted that intensity values are difficult to normalize properly, which can explain the absence of improvements in this context (Jutzi and Gross, 2009). Boulch (2020) did not measure improvements when using ConvPoint with RGB information associated with the points.

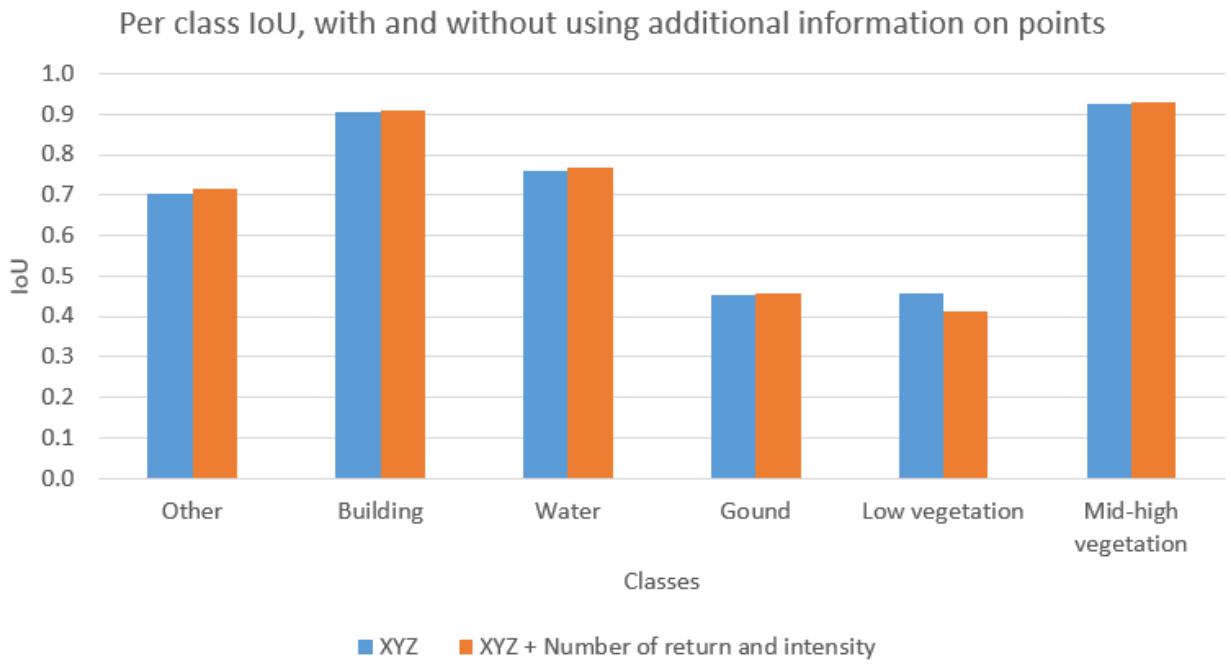


Figure 3.4 - Comparison of per class IoUs for XYZ and XYZ with the number of returns and the intensity added.

Point sampling is an important part of Boulch's method. To reduce the effect of sampling on classification results, ideally the number of points sampled should be close to the total number of

points contained in each block. The "block size" and "number of points" parameters have to be set accordingly. Table 3.5 shows the impact of block size and the number of points selected at input on the quality of classification (IoU). The best performance was obtained with 16,336 points as input. Block sizes of 35 m or 50 m gave similar results. The first value for the numbers of points (8,168) was used to maximize the GPU's memory usage and the other variations were derived from this one. The block size values tested were chosen to reduce the sampling effect, based on an average density of 14 pts/m^2, similar to the average density of our datasets.

Table 3.5 - Comparison of mean IoUs for different configurations of the number of points and block sizes.

Number of Points	Block size (m)			
	35	50	75	100
8,168	0.664	0.690	0.652	0.638
16,336	0.699	0.701	0.666	0.659
32,672	0.667	0.666	0.640	0.497
65,344	0.550	0.562	0.670	0.680

Legend
0.700
0.600
0.500

The batch size parameter was set to maximize the use of the GPU on a single Nvidia Tesla V100. For 16,336 points, the batch size was set to 12. The number of iterations depends on the size of the dataset and was assessed on the full dataset. The number of epochs was set to 50 to avoid overfitting, and has proven to be enough for models to converge. Finally, a learning rate of 0.001 was used for all the remaining tests, as other tested values did not show improvements.

3.2.6.3 Results on Full Datasets

To assess the relevance of fine-tuning models, we compared the performances of models pretrained on DALES and others trained from randomly initialized weights (Table 3.6). No improvements were noted on both datasets when using the pretraining technique.

Table 3.6 - Mean and per class IoU comparing fine-tuned and trained-from-scratch models on both datasets.

Training	Dataset	Mean IoU	IoU				
			Building	Water	Ground	Low Vegetation	Mid-High Vegetation
From scratch	Montreal	0.60	0.85	0.17	0.48	0.44	0.92
Pretrained	Montreal	0.61	0.86	0.19	0.47	0.45	0.94
From scratch	Saint-Jean	0.65	0.85	0.82	0.77	0.41	0.95
Pretrained	Saint-Jean	0.64	0.81	0.81	0.77	0.41	0.95

Models were also trained on separate and combined datasets to assess how the dataset can be generalized to other regions and topographical context. The results (IoU) on the Montreal and Saint-Jean test datasets, for different training configurations are compared in table 3.7.

Table 3.7 - Mean and per class IoU comparing different dataset configurations for training, tested on the Montreal and Saint-Jean test datasets.

Montreal Test Set						
		IoU				
Training	Mean IoU	Building	Water	Ground	Low vegetation	Mid-high vegetation
Montreal	0.60	0.85	0.17	0.48	0.44	0.92
Saint-Jean	0.33	0.73	0.09	0.29	0.03	0.80
Both	0.59	0.85	0.17	0.48	0.40	0.93

Saint-Jean Test Set						
		IoU				
Training	Mean IoU	Building	Water	Ground	Low vegetation	Mid-high vegetation
Montreal	0.38	0.69	0.45	0.18	0.03	0.91
Saint-Jean	0.65	0.85	0.82	0.77	0.42	0.95
Both	0.64	0.83	0.81	0.75	0.40	0.95

The classification performances for “Water” (0.17), “Ground” (0.48) and “Low Vegetation” (0.44) are considered poor in the Montreal test dataset (Table 3.7). In comparison, results for the “Water” (0.82) and “Ground” (0.77) classes on the Saint-Jean test dataset (Table 3.7) are largely improved. The inaccuracies in the original classification of the Montreal dataset for the “Ground” class can explain in part the results for this class. Those inaccuracies affected the training process as well as the quantitative assessment. The results for the “Building” (0.85) and “Mid-High Vegetation” (0.92-0.95) classes in both datasets are comparable with results obtained in Varney *et al.* (2020), for these classes. Training on one dataset and testing on the other adversely affects the results for most of the classes, except for “Mid-high Vegetation”.

Training and inference times are important factors to consider when using deep learning models in production. Table 3.8 shows the time spent for both operations on different configurations and compared to the resulting IoU. All tests shown here were performed on the Montreal dataset, for 50 epochs. The number of iterations per epoch is an important aspect to evaluate, as it controls the number of examples seen by the network. On a large dataset such as the one used in this

experiment, a large number of iterations are required to assure that the network has seen all possible configurations and ensure good learned features. Table 3.8 shows that 500 iterations per epoch offered practically the same results as 5000 iterations per epoch, while it was almost 5 times faster to train.

Table 3.8 - Comparison of multiple times and IoUs values for training using different numbers of iterations per epoch and inference for different step size values.

Training times							
		IoU					
Iterations / epoch	Average Time (per epoch)	Mean	Building	Hydro	Ground	Low vegetation	Mid-high vegetation
500	54 min.	0.58	0.85	0.14	0.47	0.40	0.93
1000	1h22 min.	0.59	0.86	0.15	0.48	0.41	0.92
5000	4h44min.	0.60	0.86	0.16	0.48	0.44	0.93

Inference times							
		IoU					
Step size	Average Time (per square km.)	Mean	Building	Hydro	Ground	Low vegetation	Mid-high vegetation
5 m.	2h48 min.	0.61	0.86	0.17	0.48	0.45	0.93
10 m.	45 min.	0.60	0.85	0.17	0.48	0.44	0.92
15 m.	20 min.	0.59	0.85	0.16	0.46	0.43	0.92
20 m.	10 min.	0.57	0.84	0.12	0.41	0.41	0.92
25 m.	8 min.	0.54	0.83	0.11	0.37	0.37	0.91
30 m.	6 min.	0.52	0.80	0.10	0.34	0.32	0.88

During inference, the step size controls the distance (in metres) between two consecutive classifications of points' subsets. Due to the sampling process and depending on the local density, not all points can be classified by the network. The best performances were achieved with a step size of 5 m (Table 3.8), but that small size dramatically increased the time, with an average time of almost 3 hours per square kilometre. A step size of 15 m offered the best trade-off in terms of time and accuracy. Performances started to decrease as the sizes go over 15 m, causing an augmentation of unclassified points by the network. Those unclassified points were instead assigned classification of their closest neighbour.

Visual inspection was conducted on a portion of the test data to determine the best overall model, trained on both Montreal and Saint-Jean data (Figure 3.5). The points classified as "Other" are not represented here, as those would cover most of the tile and dilute the visual representation. A

majority of the points are well-classified. In some cases, we see a better delineation of buildings in the predicted tiles, as can be seen in the upper left corner of the images on the left ((a) and (b)). In the Saint-Jean dataset, most of the "Water" points are classified as "Other", resulting in a less dense "Water" region, as shown in the upper left corner of the aerial view for Saint-Jean.

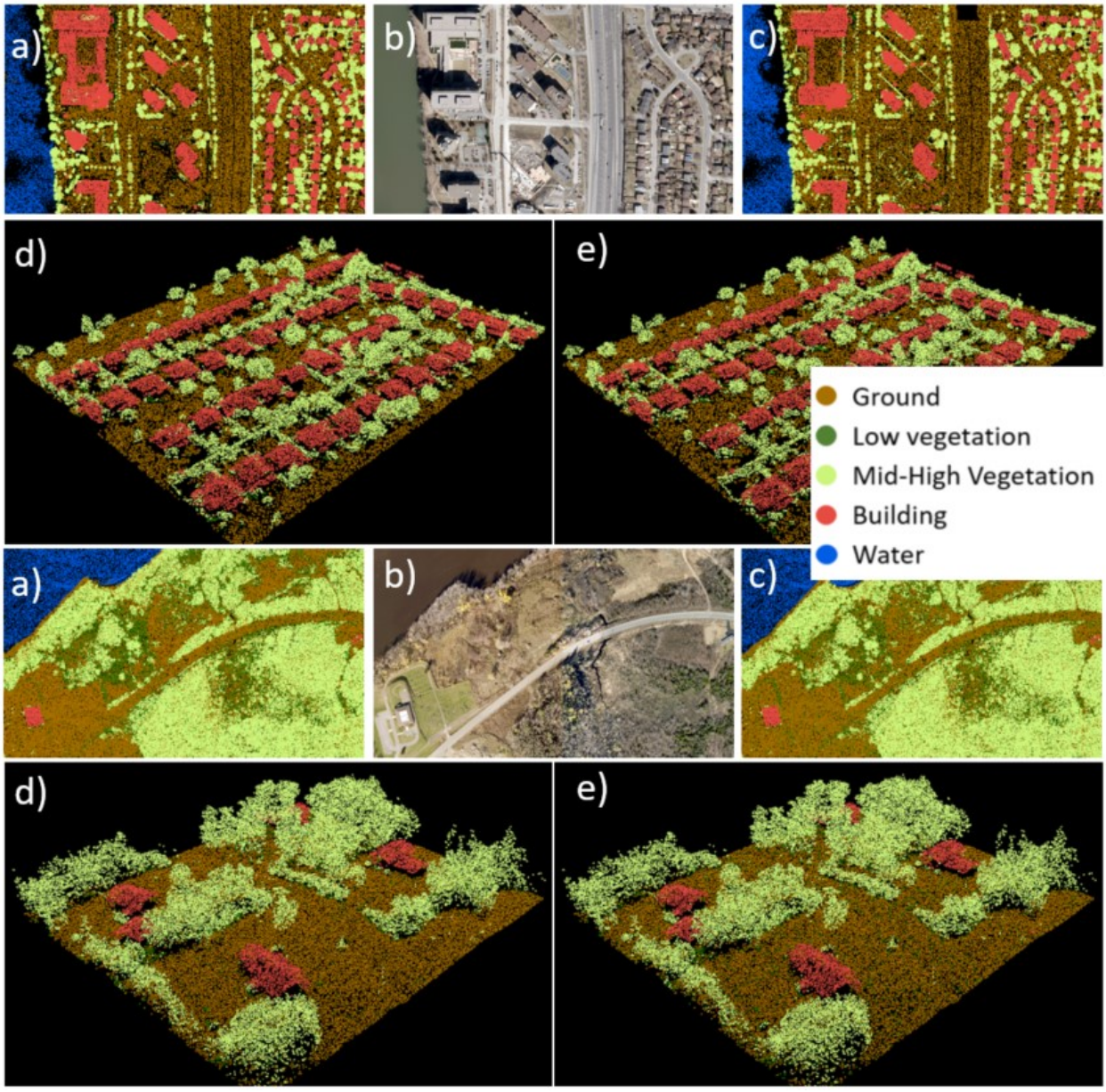


Figure 3.5 - Visual representation of the predicted classes in aerial view (a) with the corresponding imagery (b) and the ground truth (c), along with a 3D view of another area (d) and its ground truth (e). The first half of the Figure comes from the Montreal dataset and the second half is coming from Saint-Jean.

3.2.7 Discussion

The results based on the full datasets did not show improvements when the model was fine-tuned using pretraining on the DALES dataset (table 3.6). This lack of improvement could be caused by the quality and the amount of data used for training. The point density of DALES is close to 50 pts/m^2 and our datasets are around 13 pts/m^2, making the data quite different. The change in density implies that the sampled points will cover more area in a lower density point cloud, and so the features learned on the DALES dataset may not apply to a lower density context. In the DALES-trained network, learned features could be specific to the output classes, meaning that these features would be trained for poles, fences or transmitting lines, which are not relevant to our study.

Our experiment provides insights to optimize training and inference times. As we showed, there is a small trade-off between optimal performances and the processing times required to perform both operations. Improving the processing times required for both operations would reduce this trade-off. One relatively simple improvement would be to enable the use of multi-GPUs for both operations.

The quality of the original classification is very different in both Montreal and Saint-Jean datasets. Based on the dataset assessment (table 3.4) along with the IoU results for the Montreal and Saint-Jean (table 3.7) datasets, we can say that the “Ground” and “Other” classes were originally classified quite differently in each dataset. For Montreal, there are far fewer “Ground” points than for Saint-Jean, resulting in an over representation of the “Other” class for Montreal. For the “Building” class, these differences are not visible, but we can see in the results that when the network was trained on only one dataset, performances on the other dataset are greatly reduced in this class. The results also show that when trained on both datasets, the performances are about the same as when a dataset was trained and tested in the same area (i.e., trained and tested on Montreal data and trained and tested on Saint-Jean data). This suggests that the network is adapted to both contexts when trained on both datasets. In this context, this approach could be used to normalize the classification method across both datasets. These results suggest that different point cloud examples, such as multiple acquisitions in different topographic contexts, are required to improve a model’s generalization capacity.

A fixed number of points were sampled from within each block, also of fixed size. While this method makes the process computationally efficient, it also makes the learned features dependent on the sampling method. In addition, the points in LiDAR point clouds are not regularly distributed, making point density highly variable. The effect on the sampling is that blocks rarely have even close to the number of points expected, thus making the learned features unrelated to the geometry of the objects to classify. Moreover, the objects in underrepresented classes in the dataset will be poorly selected in sampling, decreasing the performance for these classes. One way to improve this situation would be to adapt the block size to the local point density. This could be achieved by calculating a local density grid at the beginning of the training, or on-the-fly during batch processing.

3.2.8 Conclusion

This experiment provides insights on the use of ConvPoint as a deep learning approach for airborne LiDAR point cloud classification on large scale urban and rural areas. The results for “Buildings” and “Mid-high Vegetation” are excellent and prove the usability of this method in a production-like environment. The differences between the Montreal and Saint-Jean datasets and the performances achieved by ConvPoint demonstrate how it can manage variabilities within the point cloud and provide usable predictions on large-scale airborne LiDAR projects.

To improve the portability and usability of the method, both the time to process and the accuracy on multiple projects have to be improved. We discussed the limitations of this approach and suggested ways to improve the computation times for both training and testing as well as to increase the prediction accuracy.

To the best of our knowledge, this is the first time such a method has been tested on this amount of data. The results and insights provided in this research have demonstrated how the efforts to accurately classify airborne LiDAR point clouds can be reduced. The high efficiency in classifying this would, at term, decrease the cost of performing the classification task, therefore increasing the usability of point clouds.

3.2.9 Acknowledgments

We would like to thank the Communauté Métropolitaine de Montreal for providing the Montreal

dataset.

4 Adaptation de la taille des blocs en fonction de la densité locale pour la sélection des points dans les nuages de points LiDAR aéroportés à grande échelle pour la méthode de classification ConvPoint (Article soumis au journal canadien de télédétection)

4.1 Résumé

Les nuages de points LiDAR aéroportés classifiés sont importants pour divers usages, notamment la cartographie des inondations et la planification urbaine. Les progrès récents dans le domaine de la vision par ordinateur, avec l'arrivée des méthodes d'apprentissage profond, ont considérablement amélioré les performances des processus automatiques pour cette tâche. L'adaptation de l'opération de convolution, traditionnellement utilisée lors de l'utilisation de réseaux neuronaux sur des images, pour la classification des nuages de points a permis d'en améliorer considérablement la précision. L'utilisation de ces nouvelles opérations de convolution doit encore être ajustée pour maximiser leur efficacité. Avec cette expérience, nous améliorons la méthode de sélection des points de ConvPoint où la sélection du sous-ensemble de points qui seront passés au réseau est adaptée à la densité locale, plutôt qu'uniforme sur toutes les scènes. Nous avons utilisé le jeu de données de référence DALES pour démontrer l'impact de la variabilité de la densité locale sur les résultats de classification de la méthode originale. La taille de bloc adaptative a conduit à certains gains de performances sur le jeu de données de référence DALES ainsi qu'à une sensibilité réduite aux hyperparamètres de la ConvPoint.

4.2 Density Dependent Block Size Adaptation for Point Selection in Large-Scale Airborne LiDAR Point Cloud for ConvPoint Classification

4.2.1 Abstract

Classified airborne LiDAR point clouds are of importance for various purposes including flood mapping and urban planning. Recent advances in the field of computer vision, with the forthcoming of deep learning methods, have drastically improved the performances of automatic processes for this task. The adaptation of the convolution operation, traditionally used when using neural networks on images, for point cloud classification has led to major improvements in accuracy. The use of these new convolution operations can still be adjusted to maximize their efficiency. With this experiment, we improve the point selection method of ConvPoint where the selection of points processed by the network is adapted to the local density, rather than uniform

across all scenes. We used the DALES benchmark dataset to demonstrate the impact of local density variability on the classification results for the original method. The adaptive block size lead to some improvements on the DALES benchmark dataset as well as a reduced sensitivity to parameterization.

4.2.2 Introduction

Airborne LiDAR point clouds are mainly used to derive digital elevation models (DEM) but this highly accurate data can also be used for various types of analysis (Yousefhussien *et al.*, 2018; Maune, 2018). To facilitate the analysis and usability of point clouds, each point needs to be classified, whereby assigned the object's class upon which the laser pulse has been reflected. The structure and non-uniformity of point clouds explain the difficulty of automating the classification task (Novac, 2018; Yousefhussien *et al.*, 2018; Zhao *et al.*, 2018, Bello *et al.*, 2020). These characteristics vary greatly from one dataset to another, depending on the acquisition specifications, the sensor used and the conditions at the time of acquisition. The capabilities offered by the various airborne LiDAR sensors also induce variability in point clouds. Another part of the variability in point clouds comes from the complex geometric structures of ground objects (Yousefhussien, 2018). The received and recorded pulses are not evenly distributed over the ground objects. Occlusions (forests, buildings, bridges, etc.) will prevent the laser pulse from reaching the ground or returning to the sensor and thus make the distribution of points in the cloud uneven. The laser signal can also be absorbed by the target on the ground, as is the case in water and sometimes rooftops (Nayegandhi and Nimetz, 2018).

The difficulties to automate this task require significant manual interventions to correct the results of the automatic classification processes, which increases the time and difficulty of preparing the LIDAR data, therefore making this task costly.

The success of deep learning algorithms in the field of computer vision has inspired several studies on point cloud classification (Huang and You, 2016; Guo *et al.*, 2019; Xie *et al.*, 2020). Recent studies have efficiently classified point cloud directly using non-discrete convolutions (Boulch, 2020; Thomas *et al.*, 2019). These methods make use of the full information in the point cloud itself thus removing unwanted transformations to the original point cloud.

Our method builds on ConvPoint (Boulch, 2020), a continuous convolution operation specifically designed to work with point cloud data. This operation has been introduced into an encoder-decoder like architecture replacing the traditional 2D convolutions with the ConvPoint operation. The ConvPoint segmentation architecture uses a fixed number of points as input, allowing easy parallelization. For large point clouds, such as those derived from airborne LiDAR, a subset of the points are randomly selected from a block of fixed size. Setting the block size and the number of points incorrectly will lead to a sampling of points unrelated to the geometric representation of the objects to classify within the block (Boulch, 2020; Varney *et al.*, 2020). Accordingly, this effect will be amplified with objects represented with fewer points in the point cloud. We propose a method to select the block size where the point selection is adapted to the local density, making it more robust to density variations within the point cloud. Using this block size adaptation, the number of points within the block and the number of points processed by the network are much closer. As a result, this improvement reduces the sampling, making learned features more related to information in the point cloud and less to the sampling method. Furthermore, we add local features such as density and block size to the points' geometric information as prior knowledge to the segmentation network proposed by Boulch (2020).

4.2.3 Related work

The two groups of methods for point cloud classification are 1) those that classify a transformed representation of the point cloud (Grilli *et al.*, 2017; Daniel, 2018) and 2) those that are using the point cloud directly, without a preprocessing step of point cloud transformation.

For the first category, point clustering is mainly used to reduce the space required for processing the point cloud (Landrieu and Simonovsky, 2018; Landrieu and Boussaha, 2019; Xie *et al.*, 2020). First attempts transformed the point cloud into 2D matrices and uses those matrices in fully convolutional networks (FCNs), largely used in semantic segmentation of images. Among these matrices are height, orientation and slope maps (Hamraz *et al.*, 2018; Zhao *et al.*, 2018). Transformations of point clouds into 2D matrices sometimes require considerable processing time and invariably lead to a loss of information, which in turn affects the quality of the classification (Boulch *et al.*, 2018; Hackel *et al.*, 2017).

Some authors use volumetric pixels, called voxels, to allow the use of FCN where 3D convolutions replace 2D convolutions. Point clouds are generally scattered and the distribution of points in space

is rarely regular. For voxels of regular size, this implies unnecessary encoding of a great number of empty voxels (Guo *et al.*, 2019; Xie *et al.*, 2020). In addition, the memory required to process voxelized space increases exponentially (cubically) as the voxel size decreases thus making voxels memory inefficient. To reduce the number of empty voxels, some authors have used an octree representation, where the voxels are of varying sizes. Empty spaces and simple geometries are represented with larger voxels, while more complex shapes are represented with smaller voxels to preserve the details of the segmented object (Tatarchenko *et al.*, 2017). Nonetheless, the transformation into voxel inevitably leads to a degradation of the information within the point cloud and classification results are accordingly affected (Landrieu and Simonovksy, 2018; Qi *et al.*, 2017a).

Landrieu and Simonovsky (2018) use a neighbourhood algorithm to group points with similar geometric characteristics (linearity, planarity, dispersion and verticality). The resulting clusters are then represented in the form of a graph, called the Superpoint Graph (SPG). The graphical representation allows describing the spatial and geometric relationship between the different objects, a characteristic that was missing in most approaches (Landrieu and Simonozsky, 2018). The classification of superpoints is performed using the PointNet method (Qi *et al.*, 2017a).

Classification methods that require a prior over-segmentation step are generally long to compute and leads to a loss of information originally contained in the point cloud. Moreover, evaluating the quality of clustered points is difficult and time-consuming, which is why it would be preferable to use a method that does not require this pre-processing step (Thomas *et al.*, 2019, Boulch, 2020).

Several authors have attempted to apply different deep learning techniques directly on the point cloud to minimize computation time and information degradation and improve classification results (Guo *et al.*, 2019; Xie *et al.*, 2020). Reviews by Xie *et al.* (2020) and Guo *et al.* (2019) report more than 50 methods based solely on deep learning since 2017. The method of Qi *et al.* (2017a), called PointNet, is the first strategy that effectively uses neural networks to perform classification directly on the point cloud (Landrieu and Boussaha, 2019; Boulch, 2020; Xie *et al.*, 2020). PointNet encodes global and local characteristics of a subset of points using a cascade of multi-layer perceptron (MLP). The major advantages of PointNet are its efficiency and simplicity. However, PointNet's performances are limited by its inability to account for spatial relationships between points (Qi *et al.*, 2017b; Thomas *et al.*, 2019; Boulch, 2020). Qi *et al.* (2017b) with

PointNet++, proposed an improved version of the solution, partially correcting the shortcomings of the first version. This second version greatly increases the complexity of the architecture, also significantly increasing processing times (Qi *et al.*, 2017b; Varney *et al.*, 2020).

Two approaches have successfully adapted the convolution operation to the point cloud, namely KPConv (Thomas *et al.*, 2019) and ConvPoint (Boulch, 2020). Both methods propose kernels of convolutions represented by points that vary in space (Thomas *et al.*, 2019; Boulch, 2020). These convolutions are then used in an encoder-decoder architecture, such as the UNet, to classify the points within the point cloud. KPConv and ConvPoint have been used on several reference data sets, including the DALES airborne LiDAR data set (Varney *et al.*, 2020). When compared on this dataset, classification performances of ConvPoint are slightly lower than those of KPConv, especially on the underrepresented classes. Although, ConvPoint is computer efficient with an easy parallelization and such efficiency is of importance when working with large point cloud datasets.

In both approaches, convolutions are applied to all entry points, unlike discrete convolutions, which is applied to a subset of the kernel size (Thomas *et al.*, 2019; Boulch, 2020). In this context, it is necessary to sample a subset of points that will be processed by the network. In KPConv, all points are selected within a defined search block. In this way, the amount of input points is variable, which complicates parallel processing (Boulch, 2020). For ConvPoint, a fixed number of points is randomly sampled, within a defined search block. In this way, parallel processing is facilitated since all subsets have the same number of points. However, if the block size and the number of points to sample within it aren't set properly, the sampling method will affect the features learned by the network, in which case they will not be representative of the geometry of the objects in the point cloud (Boulch, 2020; Varney *et al.*, 2020). To overcome this limitation, we propose a density dependent method for point selection that reduces the effect of sampling, enabling the network to learn better features and therefore improving results.

4.2.4 Density Dependent Block Size Adaptation

To accurately parameterize the ConvPoint method using large point cloud dataset, one must select a subset of points to be processed by the network, for prediction. The subset of points is defined by a block of fixed size and a fixed number of points to be sampled within the block. Given a point count, the block size value can be found using the density of points within the point cloud. Although, the density in most point clouds, especially in LiDAR derived ones, is highly variable

locally and depends on multiple factors, including the structure of the objects and the amount of overlap between swaths. Therefore, a single density value is inadequate to derive the block size value for an entire dataset. Figure 4.1 shows this density variability along with the impact of a fixed block size on the points sampled within it. The blocks are represented in red and the different positions illustrate how it can comprise a high (right), medium (center) and low (left) number of points within it, depending on the objects and points' organization in the point cloud.

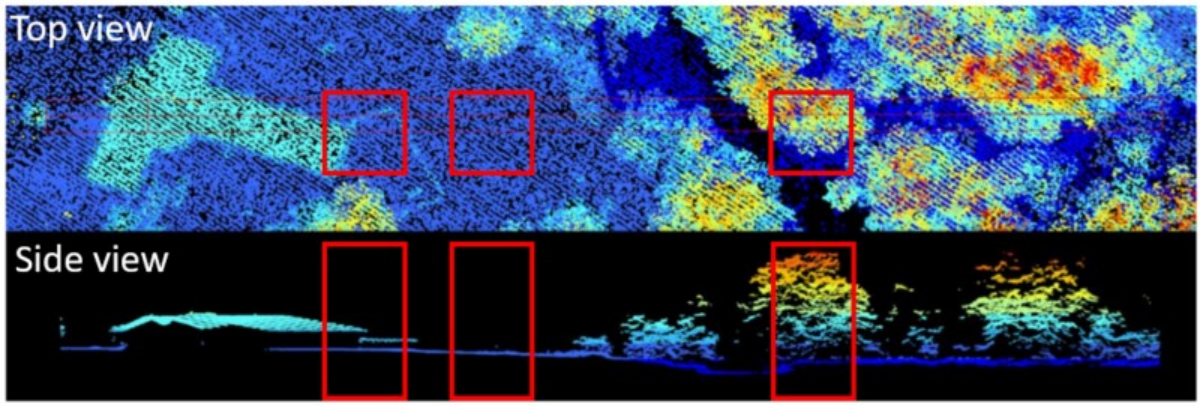


Figure 4.1 - Top and side view of a point cloud to show the points distribution and density variability.

We propose an approach to adapt the block size according to the local density and the number of points to be selected. Since the local density cannot be determined before the point selection, our approach is to select and count the number of points in a block of fixed size. The block size is then adapted according to the square root of the ratio of the number of points expected on the number of points inside the first block (Equation 4.1).

$$New\ BS = \sqrt{\frac{PE}{PI}} * \ Old\ BS$$

Equation 4.1

where *New BS* is the new block size, *PE* is the number of point desired, *PI* is the number of point within the first selection and *Old BS* is the block size used for the first point selection.

Accordingly, the total number of points within the second block will be much closer to the expected number of points, thus significantly reducing the sampling effect. Moreover, the local density, calculated as the number of points per square metre, as well as the new block size can be added to the geometric information (x, y, z) as additional input features. This block size adaptation

is done on the fly, right before passing the subset of points to the network for predictions and is performed for training, validation and inference.

4.2.5 Dataset

We use the DALES airborne LiDAR dataset to compare the performances of our improved ConvPoint method with other state-of-the-art methods. This dataset consists of 505 million points spread into nine classes, namely "Ground", "Vegetation", "Buildings", "Unknown", "Cars", "Fences", "Power Lines", "Trucks" and "Poles". The dataset is located in the city of Surrey, British Columbia, covers 10 km^2 and contains 40 tiles with an average density of 50 points/m^2 (Varney *et al.*, 2020). The classification was initially performed using hand designed feature extraction and refined manually by human annotators (Varney *et al.*, 2020). Many point cloud classification methods have been tested on this benchmark dataset, including ConvPoint, KPConv, Pointnet, Pointnet ++ and SPGraph (Varney *et al.*, 2020). Table 4.1 summarizes some characteristics from the DALES dataset.

Table 4.1 - Characteristics of the DALES dataset (derived from Varney et al., 2020).

Characteristics		**Training**	**Testing**
Average nominal density		50 pts/m^2	
Number of points (Millions)		370	136
Number of tiles (0.25 km^2)		29	11
Point %	Ground	48.11	50.74
	Vegetation	32.70	30.15
	Building	15.41	16.91
	Unknown	1.89	0.50
	Cars	0.81	0.74
	Fences	0.54	0.46
	Power Lines	0.22	0.17
	Trucks	0.20	0.11
	Poles	0.08	0.07

4.2.6 Experiments

4.2.6.1 Local Density Analysis

To emphasize the variability in point density within a point cloud, we randomly selected 5,000 points from the DALES dataset to serve as block centres and calculated the local density (i.e. number of points per area) of these blocks. This was performed for different block sizes, namely

10 m, 20 m, 30 m and 40 m. We recorded the number of points per class within each of these blocks along with the minimum, maximum, average and standard deviation for the distribution of the elevation component of the points within the block. These elements were kept to understand the relationship between the point's classification, the variability in elevation and the local density.

We conducted a similar analysis using the block size adaptation. For each of the blocks, we adapted the block size to the local density and recorded the number of points, points' classification and the Z components (min, max, average, standard deviation). We used different number of points to sample namely 8,168; 12,252; 16,336 and 20,420. The choice for these numbers of points was based on the memory limitation, described in the next sub-section. We were then able to compare the results from both analysis.

4.2.6.2 Point Cloud Classification

We used the ConvPoint architecture (Boulch, 2020) in combination with the density dependent block size adaptation on the DALES dataset to evaluate the classification performance of our method. We also compared the classification accuracy with the fixed block size, as defined in the original method.

We used the same training and inference strategy as in the original ConvPoint paper (Boulch, 2020). For training, each epoch is made of multiple iterations. An iteration is comprised of a subset of points that are processed by the network for prediction, followed by a loss calculation and an optimization pass to update the network's parameters. We used the cross-entropy for loss calculation and the Adam algorithm for optimization, as defined in the original ConvPoint (Kingma and Ba, 2014). For each iteration, the subset of points was chosen from a block, of fixed size in the original ConvPoint and of adaptive size in our method. The location of the block was randomly selected for each iteration. Enough iterations and epochs have to be performed to learn the characteristics of the training data. We performed 500 iterations per epoch and 50 epochs for all of our experiments to be able to compare them, while reducing the training time with limited impact on accuracy.

On the test set, the area covering the tiles was regularly gridded and each cell within the grid was used as a block centre and processed by the network for prediction. The parameter controlling the grid size is called "step". All points not classified by the network were assigned the classification

of their nearest neighbour. Having a lower step value avoids having a significant number of unseen points by the network. The performances were measured globally and on each class using the Intersection over Union (IoU) evaluation metric (Equation 4.2). This metric is generally used for point cloud classification (Varney *et al.*, 2020; Thomas *et al.*, 2019).

$$IoU = \frac{TP}{TP + FP + FN}$$

Equation 4.2

where TP is the true positive, FP is the false positive and FN is the false negative.

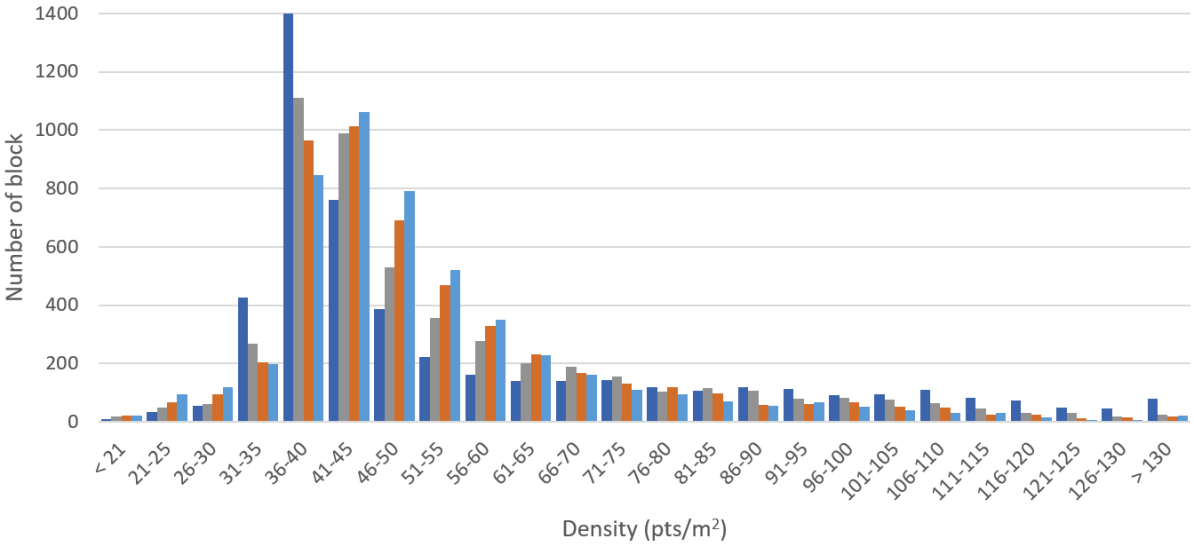
We demonstrated the impact of the "Block size" and "Number of Point" parameters in the original methods. To achieve this, we trained all combinations for 4 different "block size" (10 m, 20 m, 30 m, 40 m) and 4 configurations for "Number of Points" (8,168; 12,252; 16,336; 20,420). The choice for these numbers of points was based on the memory limitation on a single NVidia V100 GPU, to have a batch size of at least 6 (for 20,420 points). We also evaluated the impact of block size adaptation by training and evaluating models using the 4 configurations of "Number of Points" previously tested. Finally, we evaluated performances of the model when adding local features, such as local density and block size, to the input features. Those features were derived from the points within the block before being fed to the network.

4.2.7 Results

4.2.7.1 Local Density Analysis

To support our premise that the density is highly variable within the point cloud, we evaluated the local density within 5,000 randomly selected block of multiple sizes. Figure 4.2 shows the variability in local density for those 5,000 blocks, for block sizes of 10 m, 20 m, 30 m and 40 m.

The average density of the DALES dataset is said to be 50 pts/m^2 (Varney *et al.*, 2020). Although locally, the density within each block ranged from 16 to 146 pts/m^2 (20 m), with an average of 54.2 pts/m^2 and a standard deviation of 22.00 pts/m^2 for the selected blocks, supporting the premise of high variability. This evaluation also demonstrated that only 48 % of the blocks contained between 40 and 60 pts/m^2 (50 pts/m^2 ± 20 %). The analysis also showed that 14 % of the blocks had a density of 80 pts/m^2 or more. The density variability was demonstrated for all block sizes but was greater for the small block sizes (10 m, 20 m).



	10m	20m	30m	40m
Average	56.7	54.2	52.5	51.4
Standard deviation	27.2	22.0	19.6	18.2
Minimum	12	16	15	14
Maximum	170	146	143	143

Figure 4.2 - Histogram showing the variation in point density (pts/m2) for 5 000 randomly selected blocks of 18m in height and width.

We then established the effect of local density on the representation of points per class. Figure 4.3 shows the relation between the local density and the average number of points for each class, for the 5,000 blocks, for the multiple block sizes.

The representation of "Vegetation" increased drastically with the density. The number of points from the "Ground" class was relatively stable across all densities while the number of points for the "Building" class decreased with the density. All the other classes tend to have more points in blocks of lower density (< 60 pts/m^2). Blocks with a density above 60 pts/m^2 were composed at 95 % and more of ground and vegetation.

We then evaluated the relation between the point's elevation distribution and the density within the blocks. Figure 4.4 shows the relation between the average minimum, maximum and mean elevations for the 5,000 blocks, in relation to the local density. The values in Figure 4.4 are for a block size of 20 m but other block sizes gave similar results.

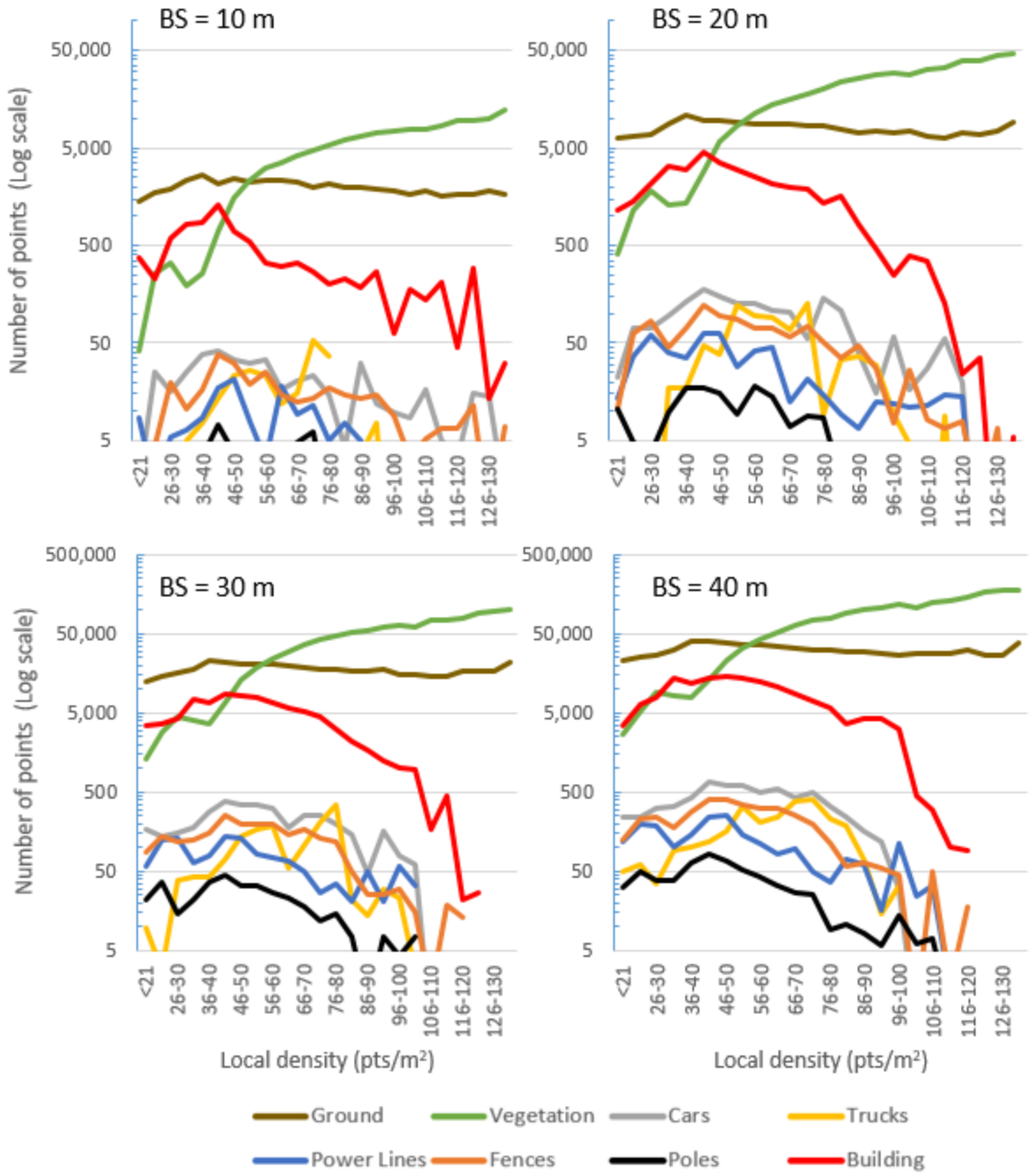


Figure 4.3 - Average number of points per class (Logarithmic scale) in relation to the local density, for multiple block sizes.

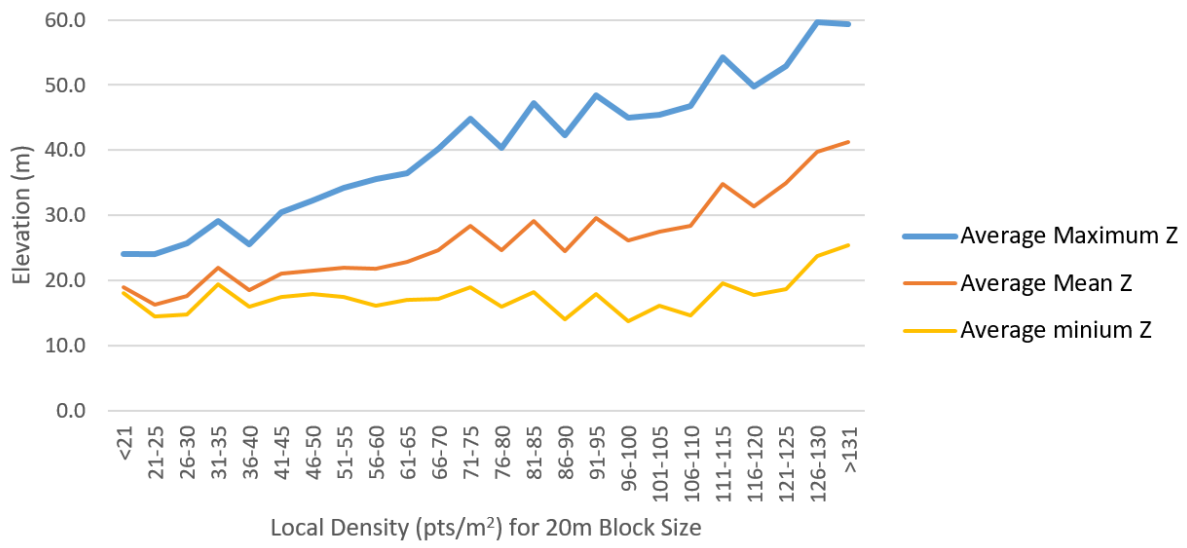


Figure 4.4 - Average minimum, maximum and mean elevation (m) in relation to the local density within blocks of 20 m.

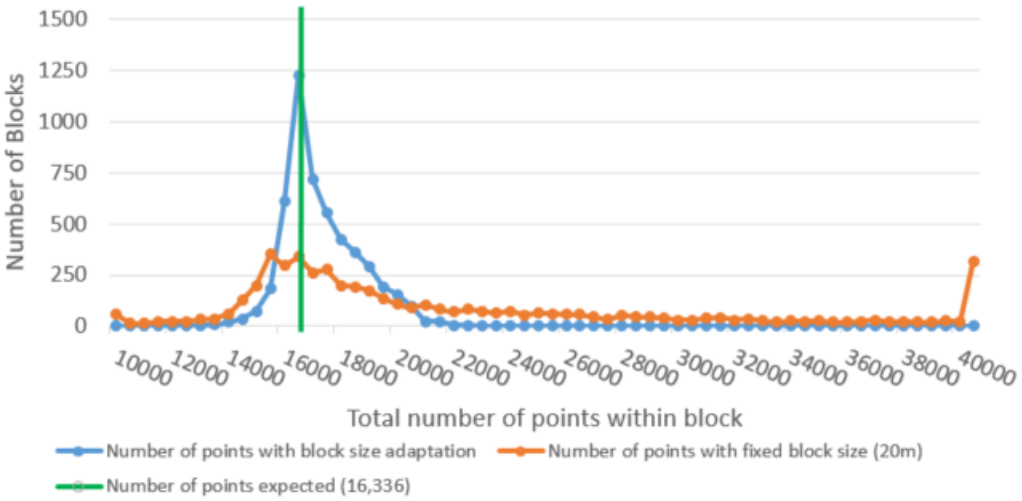
The average elevation increased with the density. The elevation range (maximum - minimum) also increased for blocks of high density. This is in agreement with Figure 4.3, where high density blocks were composed of a majority of vegetation points.

We then compared the number of points within those blocks with our method to adapt the block size. Figure 4.5 shows an example of the effect of block size adaptation on the number of points within each of the 5,000 blocks, for an expected number of points of 16,336. We added the number of points for a fixed block size of 20 m, for comparison.

For a fixed block size of 20 m, the number of points inside varied from 6,128 to 58,462. When using the block size adaptation, the number of points ranged from 12,057 to 23,336. After adaptation, the new block sizes varied from 10 m to 32 m, with an average of 18.4 m and a standard deviation of 3.1. Using the block size adaptation, 75 % of the blocks had 16,336 points ± 10 % of points, compared to 37 % with fixed block size.

4.2.7.2 Point Cloud Classification

We performed multiple training to find the best set of hyperparameters (learning rate, epochs, iterations) to compare the performances of our method with the original ConvPoint (Boulch, 2020). The results presented here shows the best performing models for each configuration. We compared the mean and per class IoU for the test dataset of the DALES dataset.



Block sizes after adaptation (m)	
Average	18.4
Standard deviation	3.1
Minimum	10.6
Maximum	32.7

Number of points per block		
	Fixed block size	Block size adaptation
Average	21,958	17,057
Standard deviation	8,896	1,350
Minimum	6,128	12,057
Maximum	58,462	23,336

Figure 4.5 - Comparison of histograms for the average number of points within blocks of fixed size of 20m (orange) and block size adaptation (blue), related to the number of points expected (green).

Both the "Block Size" and "Number of Points" parameters have impact on the learned features during the training step and accordingly affected the quality of the trained model. We trained models with variations of these parameters and compared them on the quality of classification on the test dataset. Table 4.2 shows Mean IoU values for different configurations of block size and number of points sampled.

Table 4.2 - Mean Intersection over Union (IoU) for different combinations of "Block Size" and "Number of Points".

Mean IoU				
Number of Points	**Block Size (m) (m)**			
	10	**20**	**30**	**40**
8,168	0.561	0.591	0.598	0.592
12,252	0.554	0.606	0.616	0.611
16,336	0.528	0.621	0.608	0.492
20,420	0.542	0.602	0.602	0.611

Legend
Best
Second best
Worst

These results enforce our premise that both parameters have high incidence on the network's learned features, consequently affecting the quality of the classification. The combination of a block size of 40 m and a selection of points of 16,336 has given the poorest results. Figure 4.6 shows per class IoU for different block size, given a fixed number of points sampled of 16,336. Results for the other "Number of Points" tested are provided in Annex 1.

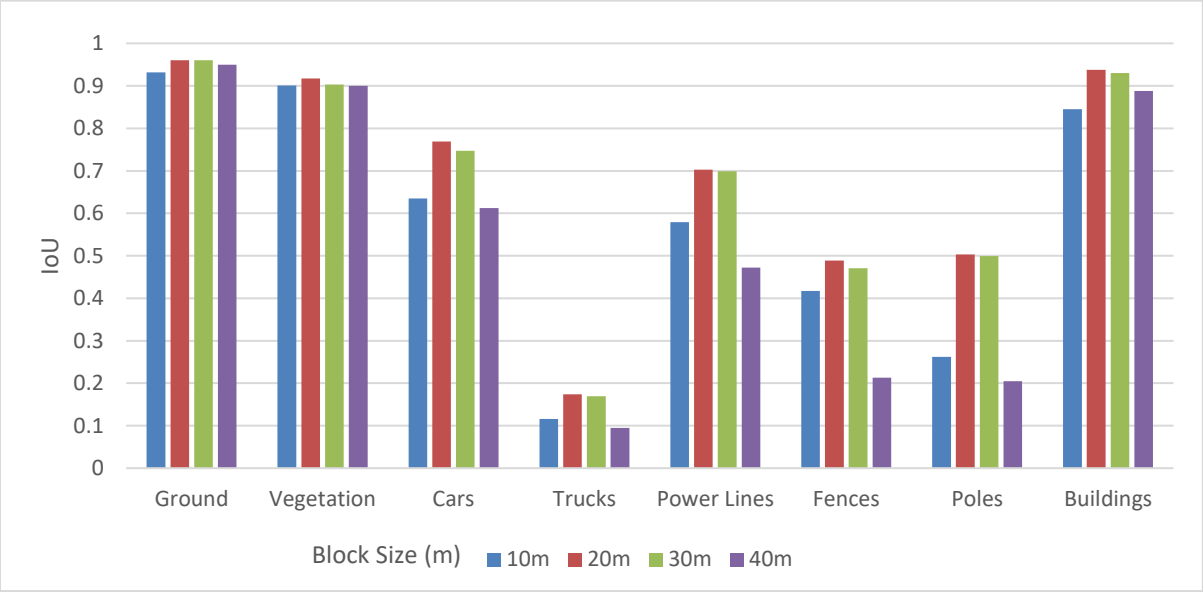


Figure 4.6 - Impact of the block size (m) on the per class IoU, with a number of selected points of 16,336.

Results showed that both parameters have limited impact on the "Ground", "Vegetation" and "Building" classes. Impacts were more significant on the "Cars", "Power Lines", "Fences" and "Poles" classes. The impact on "Trucks" were limited and results for this class were systematically low. We tested four configurations of "Number of Points" when using the block size adaptation (Figure 4.7).

These results showed that the block size adaptation reduced the sensitivity of the model to the "Number of Points" parameter, in addition to removing the block size parameter. Finally, we compared the block size adaptation with the original method. Table 4.3 shows the results for the original method and our method, as well as results from the DALES paper (Varney *et al.*, 2020). We also compared results when adding the local features (local density and block size) to the coordinates information of each point.

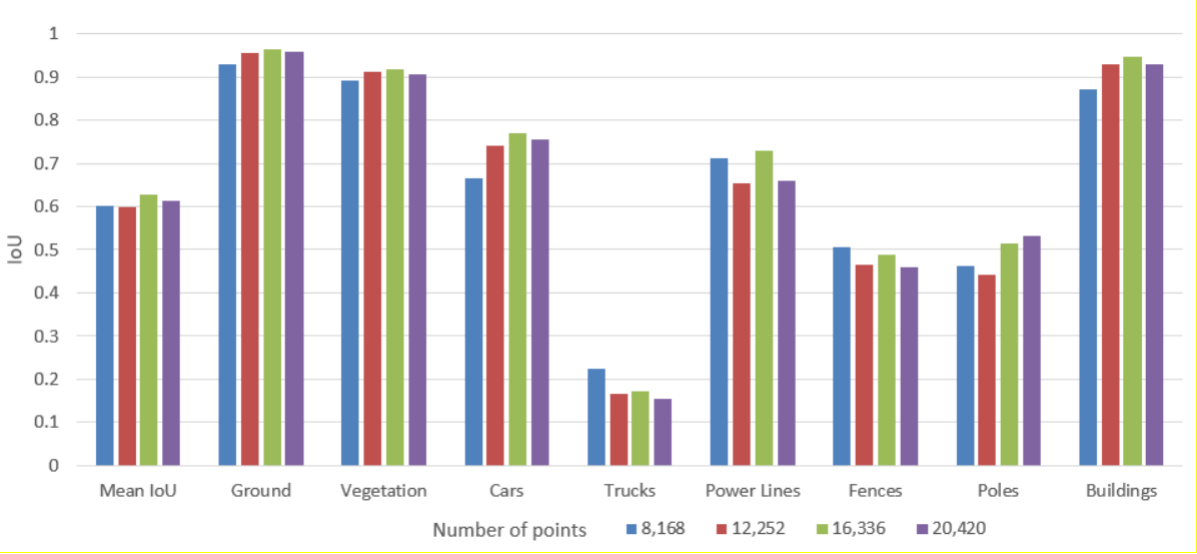


Figure 4.7 - Mean and per class Intersection over Union (IoU) using the Block Size Adaptation, for 4 configurations of number of points.

The DALES paper did not provide the hypeparameters used with all methods. Hence, we could not achieve the results presented in the DALES paper with the original ConvPoint method. Nonetheless, when we compared the original ConvPoint and our block size adaptation, our approach gave slightly better IoU for almost all classes. Adding local features did not show improvements on the classification results.

We performed a visual inspection on the test tiles to compare the results from both original and our ConvPoint. This inspection confirmed that our approach achieves slightly better classification. Some common misclassifications are located on large buildings with flat roofs. In such cases, the network would classify all points to “Ground” as it has no other context to classify otherwise. Misclassifications generally occur when there are multiple complex objects within the same area, for example dense vegetation and cars or trucks and buildings. Figure 4.8 compares the classification from both models with the corresponding ground truth, for two different regions on the test dataset.

Table 4.3 - Mean IoU and per class IoU for multiple experiments, comparing our method with the original ConvPoint along with results obtained in Varney et al., (2020).

Experiment	IoU								
	Mean	**Ground**	**Veg.**	**Cars**	**Trucks**	**Power Lines**	**Fences**	**Poles**	**Buildings**
IoU from our experiments									
Original ConvPoint (with local features)	0.615	0.955	0.916	0.750	0.156	0.719	0.471	**0.535**	0.906
Original ConvPoint (without local features)	0.621	0.960	0.917	0.769	**0.209**	0.703	0.488	0.503	0.938
Block Size Adaptation (with local features)	0.623	0.961	0.917	0.762	0.183	0.725	0.472	0.522	0.939
Block Size Adaptation (without local features)	**0.629**	**0.963**	**0.918**	**0.771**	0.171	**0.728**	**0.489**	0.515	**0.947**
IoU from Varney *et al.* (2020)									
ConvPoint	0.674	0.969	0.919	0.755	0.217	0.867	0.296	0.403	0.963
KPConv	0.811	0.971	0.941	0.853	0.419	0.955	0.635	0.750	0.966

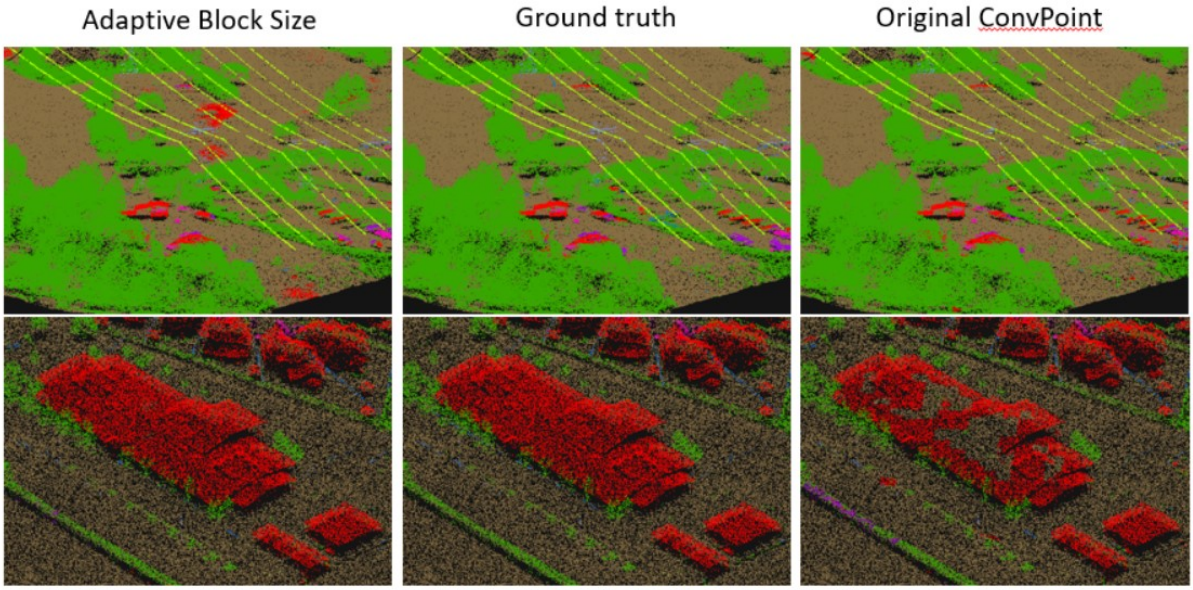


Figure 4.8 - Visual representation of classification results for our method, the original ConvPoint and the ground truth.

4.2.8 Discussion

The local density analysis has helped understand the relationship between point's distribution (density and elevation variations) and its class. The presence of vegetation allows airborne LiDAR to record points on top of the canopy as well as on the ground, which explains the increase in elevation range with density (Figure 4.3 and Figure 4.4). Accordingly, lower density blocks did

not have high proportion of vegetation points and contained relatively plane surfaces such as buildings and ground.

We demonstrated that the parameters “Block Size” and “Number of Points” have high impact on the network’s classification accuracy using the original ConvPoint (Table 4.3 and Figure 4.6). Results also suggest that 8,000 points do not provide enough geometric context to accurately capture the characteristics of the point cloud during training, therefore impacting the classification results. A similar conclusion can be drawn from the use of a 10 m block size, when using the original ConvPoint. In the original ConvPoint method, the number of points and the block size parameters should be set according to the approximate density. As our approach adapts the block size according to the local density, it removes the need to predetermine the block size. Moreover, the classification performances are less affected by the choice of the parameter “Number of Points” with our approach. This characteristic of our method reduces the burden of finding both values empirically, when using ConvPoint on other datasets.

The usage of the block size and local density as additional features to the points spatial information has not improved the classification results. Similar observations were made in the ConvPoint paper (Boulch, 2020). In the convolution operation, the feature and spatial information are aggregated using a MLP. During training, the network will learn how the aggregation will be performed. The results suggest that the training process does not consider additional information useful in ConvPoint. In comparison, the KPConv method (Thomas *et al.*, 2019) aggregates both spatial and feature information using a simple linear function, forcing the convolution to take into account the feature information. The aggregation method of ConvPoint could be improved by constraining the convolution to make use of the feature information.

4.2.9 Conclusion

The classification of airborne LiDAR data is a tedious task and improving performances of automated methods to do so is of great importance. With this experiment, we provide an improvement to the ConvPoint’s point selection method by adding a density dependent block size adaptation, whereas learned features are better fitted to the geometric characteristics of the objects within the scene. We demonstrated the effect of sampling points using a fixed block size and number of points in a reference dataset. Furthermore, classification results of our method on the DALES dataset showed some improvements for most of the classes. Overall classification

performances are still under those of KPConv and we provide insight on how to overcome this problem, while keeping the easy parallelization properties of ConvPoint. Improving the usability of deep learning methods for point classification is an important step in facilitating the analysis of LiDAR derived point cloud, leading to reduced cost for acquisition and valorization of such data.

5 Conclusion

5.1 Contribution

L'objectif principal de ce projet de maîtrise était de faciliter l'utilisation des algorithmes d'AP pour faire la classification des nuages de points issus du LiDAR aéroporté. La revue de littérature présentée dans le chapitre 2 a démontré l'intérêt de la communauté scientifique concernant la classification de nuages de points en général. Le choix de la méthode ConvPoint s'est imposé de par sa capacité à traiter plusieurs sous-ensembles de points de manière parallèle, ce qui rend la méthode rapide à utiliser et améliore la qualité des modèles d'AP suite à l'entraînement (Tatarchenko *et al.*, 2017).

L'applicabilité à grande échelle et dans différents contextes, urbains et ruraux, des méthodes d'AP sur les nuages de points LiDAR aéroportés n'avait cependant pas été réalisée. Le cas d'étude proposé au chapitre 3 de la présente recherche a démontré l'applicabilité de la méthode ConvPoint dans ce contexte. En effet, les résultats obtenus sur les jeux de données de Montréal et Saint-Jean pour les classes « Bâtiment » (0.84) et « Végétation » (0.94) démontrent l'efficacité de la méthode en milieu rural et urbain. De plus, les différentes expérimentations réalisées dans cette étude peuvent faciliter et guider la réutilisation de cette méthode dans des contextes différents.

L'amélioration proposée dans le chapitre 4, qui adapte la taille du bloc de recherche pour faire la sélection du sous-ensemble de points en fonction de la densité locale observée, a légèrement amélioré les résultats de classification, mais a surtout permis de réduire l'effort du choix des hyperparamètres de la méthode. L'étude a démontré que ces paramètres avaient, dans la méthode originale, un impact important sur la qualité de l'entraînement du modèle et par conséquent, sur les performances du modèle lorsqu'il est réutilisé en inférence. Avec notre amélioration, le paramètre de taille de bloc n'est plus requis et celui du nombre de points a maintenant très peu d'impact sur la qualité des résultats.

De manière générale, les connaissances acquises dans la présente recherche ont contribué à faciliter l'accès aux algorithmes de classification LiDAR et à terme, ces efforts peuvent réduire les coûts de valorisation de ces données.

5.2 Perspectives

Bien que les résultats présentés dans la présente recherche soient encourageants, l'amélioration des méthodes de classification doit se poursuivre.

Une limitation spécifique à la méthode ConvPoint, est que l'ajout d'information auxiliaire sur les points, comme le numéro de retour ou l'intensité, tel que testé dans la présente recherche ne permet pas d'améliorer les résultats de classification. Ces informations sont pourtant utilisées lorsque l'on effectue la classification de manière manuelle, pour faciliter la distinction entre les éléments au sol. Cette limitation peut être causée par deux facteurs, soient : la difficulté à normaliser l'intensité et l'opération de convolution ConvPoint elle-même, telle que définie par Boulch (2020). Premièrement, la normalisation de l'intensité des retours LiDAR est difficile puisqu'elle dépend des conditions d'acquisition (météo, hauteur de vol), du capteur utilisé (longueur d'onde, intensité de l'impulsion) et de la nature des cibles au sol (Jutzi et Gross, 2009). Deuxièmement, pour ConvPoint beaucoup d'importance est accordée à l'information géométrique (x, y, z), ce qui limite la contribution de l'information auxiliaire dans le calcul de la prédiction.

La sélection des points est un des éléments d'amélioration potentiels. Avec les algorithmes d'AP, seul un sous-ensemble de points peut être classifié à la fois, à cause du nombre de points et de l'espace mémoire requis pour en faire la classification. Ce découpage limite le contexte utilisé par les algorithmes d'AP pour classifier les points. Le contexte permet de décrire l'organisation des points dans l'espace et est impératif pour un résultat de classification adéquat. En fonction des classes d'éléments à classifier, le contexte qui permet d'en décrire la forme peut être de taille variable. Par exemple, le contexte nécessaire pour décrire un lampadaire est plus petit que celui pour décrire un arbre ou un édifice. L'utilisation de sous-ensembles de points de tailles multiples pourrait améliorer les résultats de classification pouvant être obtenus avec ConvPoint. L'adaptation de la méthode ConvPoint décrite au chapitre 4 permet d'adapter la taille du bloc de recherche pour la sélection des points en fonction de la densité locale observée. Cette amélioration pourrait être modifiée pour permettre une sélection de sous-ensembles à taille variable. Cette approche serait comparable aux méthodes multiéchelles, utilisées dans le traitement des images satellitaires et aériennes (Hackel *et al.*, 2016; Wang *et al.*, 2018).

Une autre limitation des méthodes d'AP, soulignée dans les chapitres 3 et 4, est la capacité de généralisation des modèles. En effet, il est difficile d'utiliser un modèle en dehors du contexte pour

lequel il a été entraîné. Cette limitation est aussi remarquée avec ConvPoint. En effet, nous avons noté qu'il est difficile d'appliquer en milieu rural un modèle entraîné en contexte urbain et vice-versa. La généralisation des modèles est cependant améliorée lorsque les données d'entraînements combinent des contextes variables. Pour atteindre un niveau de performance des modèles d'AP constant, la base de données d'entraînement du modèle doit tenir compte du caractère variable des jeux de données LiDAR, en fonction des conditions d'acquisition, des capteurs et des cibles. L'augmentation de la quantité de données LiDAR annotées pouvant être utilisée pour faire l'apprentissage de modèles est donc requise.

L'intérêt croissant pour les données LiDAR acquises par différentes plateformes (mobile, terrestre, par drone, aéroportée) contribue certainement à l'amélioration des méthodes de traitements de celles-ci. En combinant les données issues de ces multiples plateformes, on parviendra à combler les lacunes individuelles de chacune et ainsi tendre vers une gestion intégrée des données géospatiales. Cette approche holistique dans l'utilisation des données est requise pour comprendre le rôle des éléments complexes qui nous entourent et leur interaction avec l'environnement. Améliorer l'analyse et la compréhension des données LiDAR avec des méthodes de classification automatique et performante s'inscrit dans cette vision.

Références

American Society for Photogrammetry & Remote Sensing (ASPRS). (2011). LAS 1.4 Specification. *The American Society for Photogrammetry & Remote Sensing*, pp. 1–18. Récupéré au http://www.asprs.org/a/society/committees/LiDAR/LAS_1-4_R6.pdf

Armeni, I., Sener, O., Zamir, A. R., Jiang, H., Brilakis, I., Fischer, M., & Savarese, S. (2016). 3d semantic parsing of large-scale indoor spaces. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1534–1543.

Ball, J. E., Anderson, D. T., & Chan, C. S. (2017). Comprehensive survey of deep learning in remote sensing: theories, tools, and challenges for the community. *Journal of Applied Remote Sensing*, *11*(04), 1.

Bello, S. A., Yu, S., & Wang, C. (2020). *Review: deep learning on 3D point clouds*. Récupéré au http://arxiv.org/abs/2001.06280

Boulch, A. (2020). ConvPoint: Continuous convolutions for point cloud processing. *Computers & Graphics*, *88*, 24–34.

Boulch, A., Guerry, J., Le Saux, B., & Audebert, N. (2018). SnapNet: 3D point cloud semantic labeling with 2D deep segmentation networks. *Computers & Graphics*, *71*, 189–198.

Daniel, S. (2018). Revue des descripteurs tridimensionnels (3D) pour la catégorisation des nuages de points acquis avec un système LiDAR de télémétrie mobile. *Geomatica*, *72*(1), 1–15.

Dorninger, P., & Pfeifer, N. (2008). A comprehensive automated 3D approach for building extraction, reconstruction, and regularization from airborne laser scanning point clouds. *Sensors*, *8*(11), 7323–7343.

Goodfellow, I., Bengio, Y., & Courville, A. (2017). *Deep learning*. Cambridge, MA: MIT Press.

Grilli, E., Menna, F., & Remondino, F. (2017). A review of point clouds segmentation and classification algorithms. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, *42*(2W3), 339–344.

Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., & Bennamoun, M. (2019). *Deep Learning for 3D Point Clouds: A Survey*. 1–24. Récupéré au http://arxiv.org/abs/1912.12033

Hackel, T., Wegner, J. D., Savinov, N., Ladicky, L., Schindler, K., & Pollefeys, M. (2018). Large-scale supervised learning for 3D point cloud labeling: Semantic3d.net. *Photogrammetric Engineering and Remote Sensing*, *84*(5), 297–308.

Hackel, T., Wegner, J. D., & Schindler, K. (2017). Joint classification and contour extraction of large 3D point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, *130*, 231–245.

Hamraz, H., Jacobs, N. B., Contreras, M. A., & Clark, C. H. (2018). Deep learning for conifer/deciduous classification of airborne LiDAR 3D point clouds representing individual trees. *ArXiv*. Récupéré au http://arxiv.org/abs/1802.08872

Huang, J., & You, S. (2016). Point cloud labeling using 3D Convolutional Neural Network. *23rd International Conference on Pattern Recognition (ICPR)*, 2670–2675. Cancun, Mexico

Jutzi, B., & Gross, H. (2009). Normalization of lidar intensity data based on range and surface incidence angle. *International Archives of Photogrammetry and Remote Sensing*, *38*(3/W8), 213–218.

Kampffmeyer, M., Salberg, A.-B., & Jenssen, R. (2016). Semantic Segmentation of Small Objects and Modeling of Uncertainty in Urban Remote Sensing Images Using Deep Convolutional Neural Networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 680–688., Las Vegas, NV, USA.

Kingma, D. P., & Ba, J. L. (2014). Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*.

Landrieu, L., & Boussaha, M. (2019). Point Cloud Oversegmentation with Graph-Structured Deep Metric Learning. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Long Beach, CA, USA.

Landrieu, L., & Obozinski, G. (2017). Cut pursuit: Fast algorithms to learn piecewise constant functions on general weighted graphs. *SIAM Journal on Imaging Sciences*, *10*(4), 1724–1766.

Landrieu, L., & Simonovsky, M. (2018). Large-Scale Point Cloud Semantic Segmentation with Superpoint Graphs. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 4558–4567. Salt Lake City, UT, USA.

Li, Y., Bu, R., Sun, M., Wu, W., Di, X., & Chen, B. (2018). PointCNN: Convolution on X-transformed points. *Advances in Neural Information Processing Systems*, *2018-December*, 820–830.

Lin, Y., Wang, C., Zhai, D., Li, W., & Li, J. (2018). Toward better boundary preserved supervoxel segmentation for 3D point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, *143*(April), 39–47.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3431–3440. Boston, MA, USA.

Maune, D. F. (2018). DEM User Applications. In D. F. Maune & A. Nayegandhi (Eds.), *Digital Elevation Model Technologies and Applications. The DEM Users Manual.* (3rd Edition, pp. 393–436). Bethesda, Maryland: American Society for Photogrammetry and Remote Sensing.

Ministère des Forêts, de la Faune et des Parcs. (MFFP). (2021). *Imagerie et LiDAR*. https://mffp.gouv.qc.ca/les-forets/inventaire-ecoforestier/technologie-LiDAR-aerien/ Consulté le 5 mai 2021.

Nayegandhi, A., & Nimetz, J. (2018). Airborne Topographic LiDAR. In D. F. Maune & A. Nayegandhi (Eds.), *Digital Elevation Model Technologies and Applications. The DEM Users Manual.* (3rd Edition, pp. 205–236). Bethesda, Maryland: American Society for Photogrammetry and Remote Sensing.

Niemeyer, J., Rottensteiner, F., & Soergel, U. (2014). Contextual classification of LiDAR data and building object detection in urban areas. *ISPRS Journal of Photogrammetry and Remote Sensing*, *87*, 152–165.

Novac, J. M. (2018). LiDAR Data Processing. In D. F. Maune & A. Nayegandhi (Eds.), *Digital Elevation Model Technologies and Applications. The DEM Users Manual.* (3rd Edition, pp. 237–274). Bethesda, Maryland: American Society for Photogrammetry and Remote Sensing.

Özdemir, E., & Remondino, F. (2019). Classification of aerial point clouds with deep learning. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, *42*(2/W13), 103–110.

Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017a). PointNet: Deep learning on point sets for 3D classification and segmentation. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-January*, 77–85. Honolulu, HI, USA.

Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017b). PointNet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, *2017-December*, 5100–5109. Long Beach, CA, USA.

Ressources naturelles Canada (RNCan). (2017). Modèle numérique d'élévation de haute résolution (MNEHR) - Série CanÉlévation. Dans https://ouvert.canada.ca/data/fr/dataset/957782bf-847c-4644-a757-e383c0057995. Consulté le 29 avril 2020.

Ressources naturelles Canada (RNCan) & Sécurité publique du Canada (SP Canada). (2018). *Guide d'orientation fédéral sur l'acquisition de données par LiDAR aéroporté / Ressources*

naturelles Canada. Dans *General Information Product* (Vol. 117e). Gouvernement du Canada. https://doi.org/10.4095/308382

Riegler, G., Ulusoy, A. O., & Geiger, A. (2017). OctNet: Learning deep 3D representations at high resolutions. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-January*, 6620–6629. Honolulu, HI, USA.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (pp. 234–241).

Scott, G. J., England, M. R., Starms, W. A., Marcum, R. A., & Davis, C. H. (2017). Training deep convolutional neural networks for land–cover classification of high-resolution imagery. *IEEE Geoscience and Remote Sensing Letters*, *14*(4), 549–553.

Tatarchenko, M., Dosovitskiy, A., & Brox, T. (2017). Octree Generating Networks: Efficient Convolutional Architectures for High-resolution 3D Outputs. *Proceedings of the IEEE International Conference on Computer Vision*, *2017-October*, 2107–2115. Venice, Italy.

Thomas, H., Qi, C. R., Deschaud, J. E., Marcotegui, B., Goulette, F., & Guibas, L. (2019). KPConv: Flexible and deformable convolution for point clouds. *Proceedings of the IEEE International Conference on Computer Vision*, *2019-October*, 6410–6419. Seoul, Korea.

Varney, N., Asari, V. K., & Graehling, Q. (2020). *DALES: A Large-scale Aerial LiDAR Data Set for Semantic Segmentation*. Récupéré au http://arxiv.org/abs/2004.11985

Wang, L., Huang, Y., Shan, J., & He, L. (2018a). MSNet: Multi-scale convolutional network for point cloud classification. *Remote Sensing*, *10*(4), 1–20.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., & Xiao, J. (2015). 3D ShapeNets: A deep representation for volumetric shapes. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *07-12-June*, 1912–1920. Boston, MA, USA.

Xie, Y., Tian, J., & Zhu, X. X. (2020). Linking points with labels in 3D: A review of point cloud semantic segmentation. *IEEE Geoscience and Remote Sensing Magazine*, *8*(4), 38–59.

Yousefhussien, M., Kelbe, D. J., Ientilucci, E. J., & Salvaggio, C. (2018). A multi-scale fully convolutional network for semantic labeling of 3D point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, *143*, 191–204.

Zhao, R., Pang, M., & Wang, J. (2018). Classifying airborne LiDAR point clouds via deep features learned by a multi-scale convolutional neural network. *International Journal of Geographical Information Science*, *32*(5), 960–979.

Annexe 1 – Résultats détaillés pour la section 4.2.7. IoU par classe, pour différents “nombre de points”.

Mean IoU				
Number of Points	Block Size (m) (m)			
	10	20	30	40
8,168	0.56	0.59	0.60	0.59
12,252	0.55	0.61	0.62	0.61
16,336	0.53	0.62	0.61	0.49
20,420	0.54	0.60	0.60	0.61

Cars				
Number of Points	Block Size (m)			
	10	20	30	40
8,168	0.67	0.73	0.74	0.73
12,252	0.66	0.75	0.75	0.75
16,336	0.64	0.77	0.75	0.61
20,420	0.66	0.73	0.75	0.75

Ground				
Number of Points	Block Size (m)			
	10	20	30	40
8,168	0.94	0.96	0.95	0.95
12,252	0.94	0.96	0.96	0.96
16,336	0.93	0.96	0.96	0.95
20,420	0.94	0.96	0.96	0.96

Trucks				
Number of Points	Block Size (m)			
	10	20	30	40
8,168	0.13	0.12	0.16	0.16
12,252	0.13	0.17	0.18	0.20
16,336	0.12	0.17	0.17	0.10
20,420	0.13	0.16	0.18	0.15

Vegetation				
Number of Points	Block Size (m)			
	10	20	30	40
8,168	0.90	0.91	0.91	0.91
12,252	0.91	0.90	0.91	0.91
16,336	0.90	0.92	0.90	0.90
20,420	0.90	0.91	0.91	0.92

Power lines				
Number of Points	Block Size (m)			
	10	20	30	40
8,168	0.63	0.66	0.68	0.60
12,252	0.57	0.64	0.66	0.63
16,336	0.58	0.70	0.70	0.47
20,420	0.55	0.62	0.66	0.68

Fences				
Number of Points	Block Size (m)			
	10	20	30	40
8,168	0.43	0.47	0.47	0.48
12,252	0.42	0.47	0.48	0.48
16,336	0.42	0.49	0.47	0.21
20,420	0.42	0.47	0.47	0.49

Poles				
Number of Points	Block Size (m)			
	10	20	30	40
8,168	0.39	0.44	0.46	0.42
12,252	0.37	0.48	0.51	0.49
16,336	0.26	0.50	0.50	0.21
20,420	0.33	0.46	0.43	0.47

Buildings				
Number of Points	Block Size (m)			
	10	20	30	40
8,168	0.87	0.93	0.90	0.92
12,252	0.85	0.94	0.93	0.92
16,336	0.85	0.94	0.93	0.89
20,420	0.86	0.93	0.94	0.94

Legend
Best
Second best
Worst