



MASTER 2 INFORMATIQUE PARCOURS SYSTÈMES
INTERACTIFS, INTELLIGENTS ET AUTONOMES

RAPPORT DE STAGE

Segmentation sémantique d'un nuage de points et contexte

ANTOINE PÉCOUT

ENSEIGNANT RÉFÉRANT : PASCAL BALLETT

ENCADRANTS : ROMAIN CAZORLA, CÉDRIC BUCHE

7 Février 2022 - 15 Juillet 2022

Table des matières

1	Introduction	1
2	État de l’art	3
2.1	Définitions	3
2.2	Contexte scientifique	4
2.3	Méthodes orientées pointwise multi-layer perceptrons	5
2.4	Méthodes orientées recurrent neural networks	8
2.5	Méthodes orientées pointwise convolutions	9
2.6	Méthodes orientées graphes	10
2.7	Bilan	10
3	Contribution	11
3.1	Méthodes	11
3.1.1	Outils	11
3.1.2	Métriques d’évaluation	12
3.1.3	Données	14
3.2	Travaux effectués	16
3.2.1	Installation de l’environnement de travail	16
3.2.2	Prise en main de torch-points3d	17
3.2.3	Préparation des données	17
3.2.4	Analyse et présentation des résultats	17
3.2.5	Ajout d’une visualisation supplémentaire	17
3.3	Les apports du stage	18
3.3.1	Difficultés rencontrées	18
3.3.2	Compétences	19
4	Expérimentations	20
4.1	Données	20
4.2	Implémentation et Matériel	20
4.3	Résultats	21
4.3.1	Nombre de points et taille de batch	21
4.3.2	Différences d’apprentissage	22
5	Conclusion	25
6	Retour d’expérience	26

Table des figures

2.1	Exemple de segmentation sémantique sur une image du dataset ADE20K[17]	5
2.2	Illustration de la différence de structure en une image et un nuage de points.	5
2.3	Schéma de l'architecture PointNet issu de [10].	6
2.4	Schéma de l'architecture PointNet++ issu de [11]	6
2.5	Schéma de l'architecture SCF-Net(a) et du module SCF(b)[4]	8
3.1	Rendu d'une salle avec CloudCompare. À gauche la visualisation une salle brute avec les couleurs en RVB. À droite une évaluation faite par KPConv, les points corrects sont en jaune et les erreurs en rouge.	12
3.2	Illustration du calcul de Intersection over Union	13
3.3	Visualisation des areas de s3dis[1]	14
3.4	Répartition des tâches au cours du stage.	16
4.1	Évolution de la mIoU sur le jeu d'entraînement et sur le jeu de test au cours de l'entraînement. Les versions 1x1, 2x2 8k, 2x2 16k et room sont représentées respectivement en violet, orange, rose et bleu.	22
4.2	Évolution de l'accuracy sur le jeu de test au cours de l'entraînement. Les versions 1x1, 2x2 8k, 2x2 16k et room sont représentées respectivement en violet, orange, rose et bleu.	23
4.3	Évolution de l'IoU par classe au cours de l'entraînement de la version 1x1 (en haut à gauche), 2x2 8k(en haut à droite), 2x2 16k (en bas à gauche) et room (en bas à droite).	24
4.4	Matrice de confusion produite sur le jeu de test. Les versions 1x1, 2x2 8k, 2x2 16k et room sont représentées respectivement en violet, orange, rose et bleu.	24

Chapitre 1

Introduction

Pour de nombreux domaines actuels comme la conduite autonome, la robotique mobile et la réalité augmentée/virtuelle, la compréhension des scènes en 3 dimensions est nécessaire. La segmentation sémantique est une tâche classique pour la compréhension de scènes. Elle consiste à attribuer une étiquette à chaque entité. Pour une image, une entité est un pixel. Un des formats de capture courant des scènes en 3 dimensions est le nuage de points. Dans ce cas on veut attribuer un label à chacun des points. Ce type de tâche est réalisé grâce à l'apprentissage profond, une branche de l'intelligence artificielle. L'apprentissage profond, ou deep learning en anglais, a pour objectif de faire apprendre à un algorithme des motifs récurrents dans les données sur lesquelles on va l'entraîner pour qu'il puisse faire des prédictions sur des nouvelles données. Les algorithmes utilisés sont basés sur des réseaux de neurones artificiels.

Pour modifier une installation industrielle, il est souvent nécessaire de posséder sa modélisation 3D. Cependant les plans ne sont pas toujours disponibles et sans eux il faut reprendre les mesures sur place. Cette tâche est longue et peut être dangereuse. Segula Technologies souhaite développer un système pour obtenir la modélisation 3D d'une installation industrielle plus rapidement et de façon moins risquée. La segmentation sémantique de nuages de points capturés est une étape cruciale du projet. C'est sur quoi se porte la thèse de Romain CAZORLA. C'est dans le cadre de cette dernière que le sujet de stage "Segmentation sémantique d'un nuage de points et contexte" a été proposé. L'objectif est d'étudier l'impact du contexte, par exemple la propagation de l'information contextuelle ou la différence entre le contexte proche ou éloigné, sur la segmentation sémantique de nuages de points. Mes missions comportaient tout d'abord l'installation et la prise en main des outils, puis le traitement d'un jeu de données pour le décliner en plusieurs versions. Elles vont permettre d'entraîner plusieurs architectures et d'après les évaluations, pouvoir apporter des réponses à propos de l'impact de l'échelle des échantillons de données sur la segmentation sémantique

d'un nuage de points.

Ce stage de fin d'études est réalisé dans le cadre du Master 2 Systèmes Interactifs, Intelligents et Autonomes à l'Université de Bretagne Occidentale. Mon projet professionnel étant de travailler dans le domaine de la conduite autonome, le sujet de la segmentation sémantique de nuages de points est parfaitement en adéquation avec celui-ci. J'ai hésité avec un autre stage dans une entreprise de conseil. Ce dernier m'aurait permis de faire un premier pas dans une entreprise privée, cependant, le sujet, la classification de CVs par traitement du langage naturel, était moins pertinent par rapport à mon projet et de ce fait, moins motivant.

Pendant ce stage je m'attendais à approfondir mes connaissances dans le domaine du deep learning, autant au niveau de l'utilisation des différents outils que de la compréhension du fonctionnement des architectures. C'était aussi une occasion pour moi d'appréhender ce qu'est un travail de recherche sur plusieurs mois. Hésitant pour la suite, entre continuer ma formation avec une thèse et commencer à travailler en entreprise, cela m'a permis de faire mon choix.

C'est à l'École Nationale d'Ingénieurs de Brest (ENIB) que se déroule le stage, du 7 Février 2022 jusqu'au 15 Juillet 2022. Plus précisément au Centre Européen de la Réalité Virtuelle (CERV), une plateforme de recherche scientifique de l'ENIB. Environ 40 chercheurs y travaillent sur la réalité virtuelle associée à 4 thèmes principaux : systèmes complexes, formation et patrimoine, agents autonomes et les arts.

Ce document débute par la présentation du contexte scientifique et d'un état de l'art de la segmentation sémantique de nuages de points. Il aborde ensuite le déroulement du stage, les différents outils utilisés, les travaux réalisés et apports, pour enfin présenter les expérimentations et conclure sur une réponse à la problématique.

Chapitre 2

État de l'art

Le résumé de l'état de l'art commence par la définition de termes importants quand on traite d'apprentissage profond puis présente les grandes étapes de l'avancée des techniques de segmentation sémantique de nuages de points.

2.1 Définitions

Caractéristique / Feature

Une caractéristique une valeur mesurable associée à une entité qui va permettre au modèle de machine learning de reconnaître des motifs pour faire des prédictions.

Perceptron Multi-Couches / MultiLayer-Perceptron / MLP

Un perceptron est aussi appelé neurone artificiel. Sa sortie s est calculée comme :

$$s = f\left(\sum_{i=0}^k w_i e_i + b_i\right) \quad (2.1)$$

Où k est le nombre d'entrées, e_i , b_i et w_i respectivement la valeur, le biais et le poids pour l'entrée i et f une fonction d'activation. Un MLP est un réseau de neurones artificiels organisés en couches. Chaque neurone d'une couche a pour entrée toutes les sorties de la couche précédente.

Réseau de Neurones Récurrent / Reccurent Neural Network / RNN

Un réseau de neurones récurrent est un réseau de neurones doté d'une mémoire à court ou long terme grâce à des couches dont les entrées proviennent de couches situées après elles.

Mise en commun / Pooling

Le pooling permet d'agréger plusieurs entités dans une seule grâce à une opération. Par exemple avec un maxpooling, pour chaque feature on sauvegarde seulement la valeur maximale dans le groupe.

k Voisins les plus proches / k-Nearest Neighbors / kNN

L'algorithme des kNN sélectionne les k points les plus proches d'un point défini avec $k \leq N$.

Échantillon / Sample

Un échantillon est l'élément passé en entrée d'un réseau de neurones.

Taille de Batch / Batchsize

Le batch est l'entité qui va être prise en compte pour chaque optimisation du modèle. Sa taille définit le nombre d'échantillons qu'il contient.

Epoch

L'epoch est l'unité de mesure du temps d'entraînement. Une epoch est un passage complet du jeu de données d'entraînement dans le modèle.

2.2 Contexte scientifique

La segmentation sémantique est une tâche courante en traitement d'image. Elle consiste à assigner à chaque pixel un label. Un exemple sur une image du jeu de données ADE20K[17] est visible en Figure 2.1 Les nuages de points et les images ont des structures radicalement différentes : Dans une image les pixels sont rangés par lignes et par colonnes tandis que le nuage de points est un ensemble non structuré et non ordonné, illustré par la Figure 2.2. Il est donc impossible d'appliquer aux nuages de points les mêmes transformations qu'aux images[10]. Il a donc fallu trouver des méthodes pour surpasser ou contourner cette difficulté.

Les premières méthodes de segmentation sémantique de nuages de points se tournaient vers la projection des nuages de points dans des ensembles ordonnés. Une première approche était de créer des images du nuage vu sous différents angles pour ensuite pouvoir analyser l'ensemble d'images[2]. Cette méthode a l'inconvénient de faire beaucoup d'approximations. D'autres projetaient le nuage sur une grille en 3 dimensions. En fonction de la taille de la grille ces méthodes pouvaient elles aussi faire beaucoup d'approximations. Par exemple, si

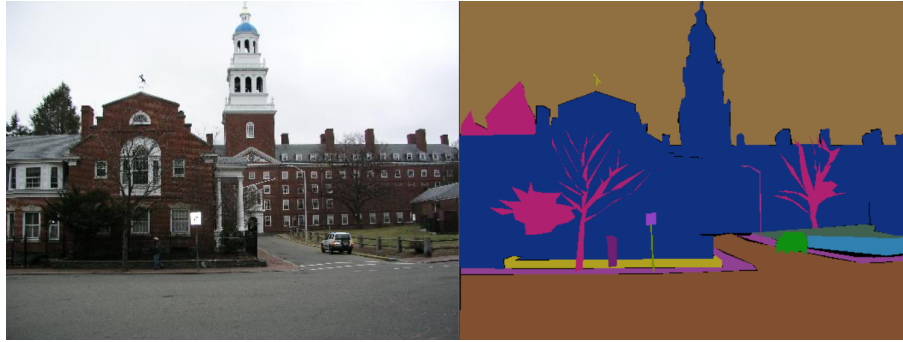


FIGURE 2.1 – Exemple de segmentation sémantique sur une image du dataset ADE20K[17]

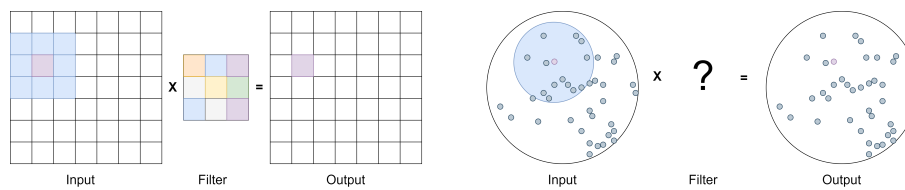


FIGURE 2.2 – Illustration de la différence de structure en une image et un nuage de points.

deux points se trouvent dans la même cellule : comment les différencier ? Cette projection est également très gourmande en ressources de calculs[12].

Les nuages de points étant des ensembles non-ordonnés, les méthodes travaillant directement sur ces derniers se doivent d'être invariantes à l'ordre des points. Ces dernières se concentrent sur la représentation des points et de leurs informations contextuelles pour ensuite apprendre des caractéristiques grâce aux Multi-Layer Perceptrons (MLPs). On peut voir différentes variantes émerger comme l'ajout de réseaux de neurones récurrents ou des adaptations de la convolution. D'autres méthodes utilisent les graphes pour mieux structurer le nuage.

2.3 Méthodes orientées pointwise multi-layer perceptrons

La première méthode à avoir réussi à travailler directement sur les nuages de points est PointNet[10]. Dans cette architecture, les points sont passés dans plusieurs couches de MLPs qu'ils partagent pour extraire les caractéristiques locales du nuage. Puis les caractéristiques locales sont agrégées par le biais d'un maxpooling pour obtenir les caractéristiques globales. Pour de la classification, on passe les caractéristiques dans un dernier MLP pour prédire la classe. En revanche, pour la segmentation sémantique, les caractéristiques locales et globales sont concaténées. Puis elles sont passées dans un premier MLP qui va sortir les caractéristiques pour chaque point, puis dans un second pour attribuer une classe à chaque

point. Un schéma de l'architecture est visible en Figure 2.3.

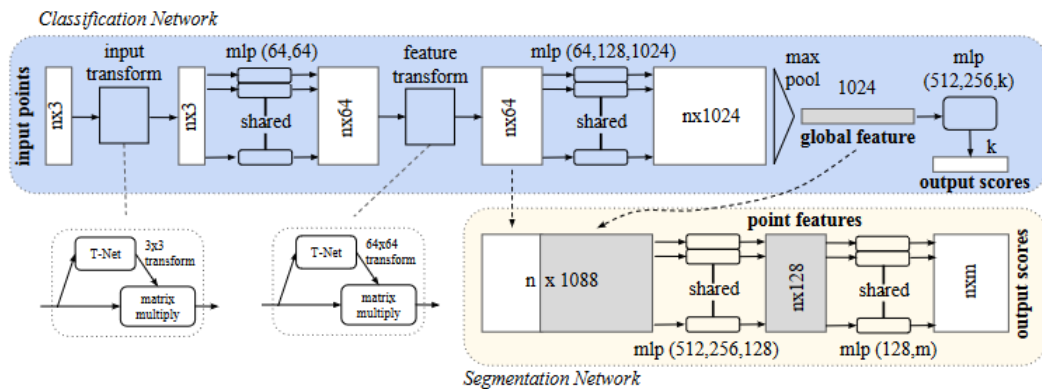


FIGURE 2.3 – Schéma de l'architecture PointNet issu de [10].

PointNet[10] a pour problème de ne pas bien capturer la structure locale du fait que la totalité de l'agrégation soit faite par un seul maxpooling. C'est pour y remédier qu'a été présenté PointNet++[11]. L'idée apportée ici pour appréhender la structure locale est d'utiliser une architecture encoder-decoder de type U-Net pour baisser la résolution de nuage tout en agréant un contexte local de plus en plus grand. Les réseaux de type U-Net se composent de 3 types de modules : Le downsampling qui encode les caractéristiques, le module intérieur pour transformer les features apprises et enfin le module d'upsampling pour propager les features. Un schéma de l'architecture de PointNet++ est visible en figure 2.4.

Le module de downsampling est appelé ici "Set Abstraction" (SA). La sélection des points sous lesquels regrouper les caractéristiques se fait par l'algorithme "Farthest Point Sampling" pour recouvrir au mieux le nuage. Deux modules permettant d'extraire des caractéristiques sont proposés. Premièrement, le "Multi-Scale Grouping" ou MSG qui va

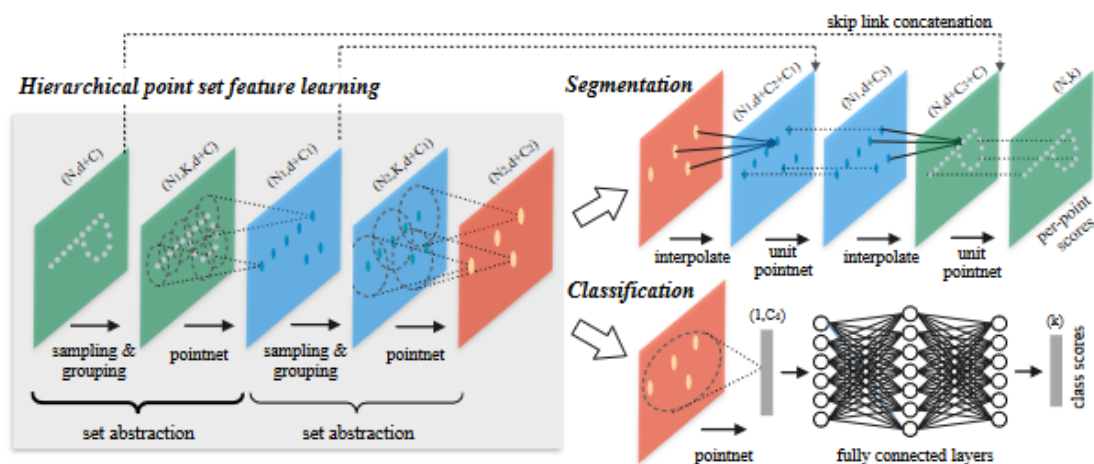


FIGURE 2.4 – Schéma de l'architecture PointNet++ issu de [11]

agréger les features des points dans un rayon défini autour d'un super point. PointNet[10]. Grouper des points pour différents rayons permet d'apprendre à différentes échelles. Les caractéristiques des différentes échelles sont concaténées pour créer les caractéristiques du super-point. Deuxièmement, il y a le "Multi-Resolution Grouping" ou MRG. Moins coûteux en calcul, la caractéristique renvoyée est séparée en deux parties. La première est la somme des features apprises à l'itération précédente pour cette région (super-points précédents présents dans le rayon de l'actuel) et la deuxième est apprise en passant tous les points de l'ensemble initial dans un PointNet. Pour les deux couches, la sélection des points par région permet d'outrepasser les problèmes dus à la densité non uniforme des nuages de points. Pour attribuer un label à chaque point de l'ensemble, après le down-sampling vient les différents passages par le module "Feature Propagation" (FP). Les caractéristiques sont propagées par interpolation avec les voisins les plus proches et pondérées en fonction de la distance métrique.

Avec PointNet++[11] les caractéristiques locales sont agrégées par maxpooling et cela crée une perte d'informations car seul la valeur maximale est gardée. [7] propose un nouveau module pour la description locale appelé PointSIFT. Les modules SA et FP présentés dans [11] ont été repris et modifiés. L'objectif du module PointSIFT est d'encoder pour chaque point les informations des voisins les plus proches selon leur orientation relative au point. Le module se compose de trois unités d'Orientation Encoding (OE). Une unité OE récupère les informations du voisin le plus proche dans 8 directions pour former un cube $2*2*2$. L'unité va ensuite appliquer des convolutions successivement sur les axes X, Y et Z pour retourner un vecteur. Chaque unité du module va avoir une portée différente, ce qui va permettre d'extraire les caractéristiques à différentes échelles. Les vecteurs des trois échelles sont ensuite concaténés et une convolution est appliquée pour obtenir la feature retournée par le module.

[16] propose une autre approche pour représenter la structure locale. Le module "Adaptive Feature Adjustment" (AFA) se concentre sur les interactions entre les points d'une même région en créant un graphe complet. Ces graphes permettent de créer de nouvelles caractéristiques pouvant être apprises par les MLPs. Le module est intégré à la partie SA de l'architecture de PointNet++[11].

Le sampling le plus courant était le FarthestPointSampling, cependant [5] a montré que le RandomSampling (sélection aléatoire des points) reste efficace tout en ayant une complexité bien inférieure, $O(N^2)$ pour le FarthestPointSampling contre $O(N)$ pour le RandomSampling, ce qui le rend intéressant pour pouvoir travailler sur des grands nuages de points. Au moment de sa publication, RandLA-Net[5] peut traiter des nuages de l'ordre du millions de points avec des performances égalant ou dépassant l'état de l'art sur S3DIS[1]. Le module

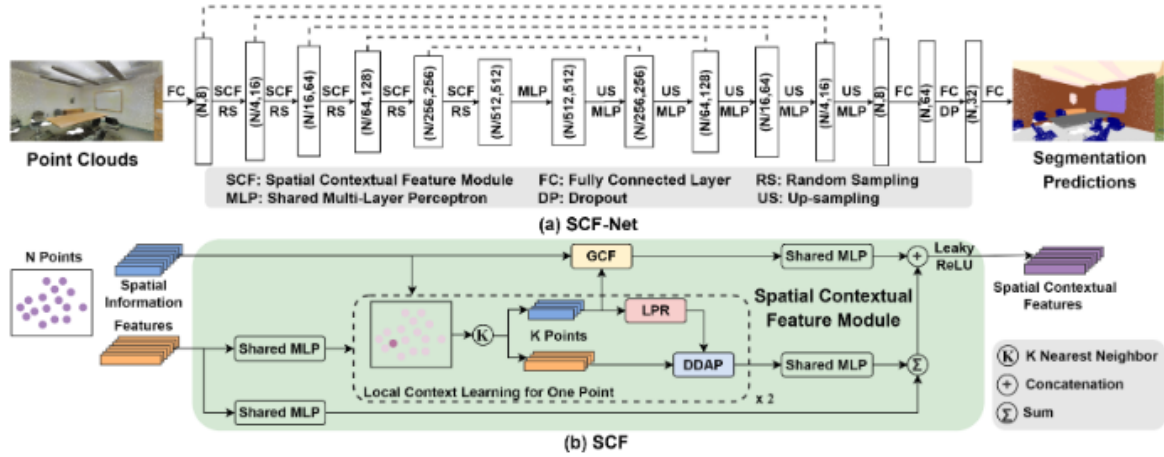


FIGURE 2.5 – Schéma de l’architecture SCF-Net(a) et du module SCF(b)[4] .

de downsampling de RandLA-Net utilise le RandomSampling et des DilatedResidualBlocks pour augmenter progressivement la portée des features tout en gardant une faible complexité.

Basé sur RandLA-Net, SCF-Net[4] utilise aussi le RandomSampling pour pouvoir traiter des scènes à grande échelle. Le module SpatialContextualFeature (SCF)[4] crée trois représentations du nuage de points. D’abord pour chaque point, il y a une représentation local polaire calculée en fonction de ses k-Nearest Neighbors (kNNs), c’est le module Local Polar Representation (LPR). Elle permet une invariance à la rotation autour de l’axe vertical. Puis le module Dual-Distance Attentive Pooling (DDAP) représente l’information contextuelle à partir des distances géométriques entre le point et ses kNNs et des features apprises par un MLP. Enfin, le module Global Contextual Feature (GCF) a pour but de créer une feature contextuelle couvrant une plus grande zone. Ces trois représentations sont concaténées et passées dans des MLPs partagés de façon similaire aux architectures précédentes. Un schéma de ce module et de l’architecture présentée par [4] est visible en Figure 2.5.

2.4 Méthodes orientées recurrent neural networks

Dans [15] deux nouvelles composantes sont présentées. Les ”Pointwise Pyramid Pooling” (3P) pour représenter les informations locales et des Recurrent Neural Networks (RNNs) à travers lesquels les points sont passés dans un ordre régité par des fenêtres glissantes pour diffuser l’information contextuelle à longue distance. Au début, l’ensemble des points passe par un MLP partagé pour apprendre les caractéristiques des points. Ensuite, pour le module 3P, l’espace est séparé en blocs de 1.5m x 1.5m couvrants toute la hauteur de la pièce. Ces blocs sont encore divisés pour créer des sous-espaces de différentes échelles. Le 3P utilise un maxpooling sur les features apprises d’un nombre N de points aléatoires dans un même bloc.

Les caractéristiques des différentes échelles sont concaténées pour obtenir la feature locale. Pour la partie avec les RNNs, l’espace est aussi divisé en blocs mais avec une longueur et une largeur non uniformes. Ensuite, tous les blocs avec un même index en x sont passés dans une unité RNN indépendante. Les blocs de sortie sont assemblés puis re-divisés de la même façon. Enfin, les blocs partageant un même index y sont passés dans d’autres unités RNN indépendantes. Pour finir, les caractéristiques des points, celles apprises par le module 3P et celles apprises par les RNNs sont concaténées et passées dans un MLP pour obtenir la classification. D’après les auteurs [15] cette méthode a des difficultés pour différencier les classes sémantiques ayant des formes proches comme les murs et les portes.

2.5 Méthodes orientées pointwise convolutions

La convolution est très efficace pour révéler des caractéristiques dans une image. On a alors voulu adapter cette opération aux nuages de points. L’idée apportée par [6] est assez ressemblante à une convolution sur des voxels : le point traité est placé au centre d’une grille $3 \times 3 \times 3$. Cette grille fera office de filtre de convolution. La convolution peut être écrite comme sur la Formule 2.2. Où k itère sur les cellules de la grille et $\Omega_i(k)$ est la k -ième cellule du filtre centré sur le point i dont les coordonnées sont p_j . j permet de compter les points d’un sous-domaine. w_k est le poids associé à la cellule. les valeurs features des points i et j sont x_i et x_j . l et $l - 1$ représentent l’index de la couche du réseau.

$$x_i^l = \sum_k w_k \frac{1}{|\Omega_i(k)|} \sum_{p_j \in \Omega_i(k)} x_j^{l-1} \quad (2.2)$$

KernelPoint Convolution (KPCConv), présentée par [13], est plus adaptée aux nuages de points. Les poids du masque de convolution ne sont pas portés par les cases d’une grille 3D mais par des points répartis avec une forme régulière autour du point sous lequel regrouper les caractéristiques. C’est KPCConvRigid. La version KPCConvDeformable permet aux kernel points de se déplacer pour s’adapter au nuage. Chaque kernel point diffuse son poids aux points du voisinage dans un certain rayon. La position et le champs d’action des KernelPoints sont définis de manière à ce que les champs se chevauchent, la fracture est moins brusque qu’avec une grille 3D. Ici on considère les coordonnées des points comme la structure du nuage et les features comme les vraies données. La position n’est utile que pour appliquer les poids de la convolution. Le pipeline complet reprend aussi une architecture encoder-decoder. Le downsampling se fait par gridsampling, en augmentant la taille de la grille au fur et à mesure des couches. Au moment de la publication, en 2019, les deux versions avaient des résultats égalant voire dépassant l’état de l’art. Cependant, KPCConvRigid est meilleure sur

les tâches simples comme la classification ou des petites segmentations tandis que KPConv réussit mieux sur les tâches complexes comme des segmentations avec un nuage important ou des objets diversifiés[13].

2.6 Méthodes orientées graphes

Certaines méthodes essaient de capturer la structure locale créant des graphes. Pour chaque point, Dynamic Graph CNN[14] construit un graphe avec ses voisins par le biais des k-Nearest Neighbors pour ensuite appliquer une transformation semblable à une convolution. Les auteurs ont nommé ce module EdgeConv. [9] propose une approche différente. L'idée est de sur-segmenter le nuage de points manière non-supervisée. La sur-segmentation va permettre de créer des groupes de points avec des formes simples et supposés du même label. Les super points adjacents sont reliés par des super edges. L'architecture PointNet[10] est ensuite utilisée pour extraire les caractéristiques des super points. Enfin les labels sont calculés par des Gated Recurrent Units auxquels les super points adjacents sont reliés. [8] présente une amélioration de la sur-segmentation et de la représentation locale des points.

2.7 Bilan

PointNet[10] et PointNet++[11] ont posé les bases des méthodes travaillant directement sur les nuages de points. Comme vu dans la partie précédente, la plupart des méthodes utilisent une architecture U-Net. Elles se concentrent sur la représentation des points d'un contexte local, relative ou polaire, et sur le moyen d'encoder les features pour augmenter leur dimension et diminuer la résolution du nuage. Les convolutions orientées points essaient de tirer un avantage supplémentaire des coordonnées de points, en plus prendre en compte la région locale, l'orientation et la position dans la région ont de l'importance. KPConv va même jusqu'à considérer les coordonnées comme la structure du nuage et ne les utilise que pour appliquer la convolution.

Dans les publications vues précédemment, certaines méthodes découpent le nuages en colonnes de 1m x 1m comme [10] et [16], d'autres utilisent les salles entières comme [5], [15] ou [7]. [13] découpe le nuage d'entraînement en sphères si le matériel ne permet pas de le traiter d'un coup avec KPConv. Malgré ces méthodes variés, il n'y a pas d'étude les comparant et pointant leurs points forts ou faibles. C'est cet aspect qui a été travaillé pendant ce stage.

Chapitre 3

Contribution

3.1 Méthodes

Travailler avec des réseaux de neurones nécessite d’avoir des données sur lesquelles les entraîner et des métriques pour les évaluer. Cette partie présente les différents outils, métriques et données utilisés.

3.1.1 Outils

Les outils suivants ont servi à faciliter ce stage de la mise en place des modèles jusqu’à leur évaluation en passant par l’entraînement.

Torch-points3d

Il peut être compliqué de développer et tester des modèles de deep learning qui travaillent sur les nuages de points. C’est pour faciliter ces tâches qu’a été développé torch-points3d[3], basé sur PyTorch. C’est le seul framework permettant une gestion facilitée des données, de l’entraînement et l’évaluation des modèles. Quatre tâches sont implémentées : Segmentation sémantique, classification, superposition et détection d’objets. Plusieurs réseaux de la littérature sont déjà intégrés, ainsi que des architectures de base pour aider au développement et à la modification de réseaux. Grâce à Hydra¹ et aux fichiers de configuration, la plupart des paramètres sont modifiables rapidement, que ce soit pour l’architecture, le dataset, l’entraînement ou l’évaluation.

1. hydra.cc

Weights and Biases

Connecté à torch-points3d, Weights and Biases² (Wandb) permet de visualiser l'avancement des entraînements en temps réel. Pour chaque entraînement, Wandb affiche en ligne les courbes de plusieurs valeurs en fonction du nombre d'epochs comme des métriques mais aussi des informations matérielles : température des composants, utilisation de ressources. Cela permet de mieux comprendre comment le modèle s'améliore au cours de l'entraînement et éventuellement identifier des problèmes.

CloudCompare

Pour visualiser les résultats des segmentations, un outil permettant de visualiser, modifier et coloriser les nuages de points était nécessaire. C'est ici qu'intervient CloudCompare³ qui de plus est open source. Le logiciel possède une fonction qui permet de recolorer les nuages de points en fonction de valeurs. On peut par exemple calculer la différence entre le label prédit et la vérité et colorier les points d'une couleur si c'est égal à zéro et d'une autre dans le cas contraire. À partir de cette valeur et le label d'origine on peut déduire la classe avec laquelle le point a été confondu. Sur la Figure 3.1 on peut voir que la porte(7) et la colonne(6) ont été confondues avec un mur(11).

3.1.2 Métriques d'évaluation

Pour quantifier les performances d'un modèle et l'évaluer de manière objective, on utilise des métriques. Cette partie présente les différentes métriques utilisées dans ce document.

-
2. wandb.ai/site
 3. www.danielgm.net/cc

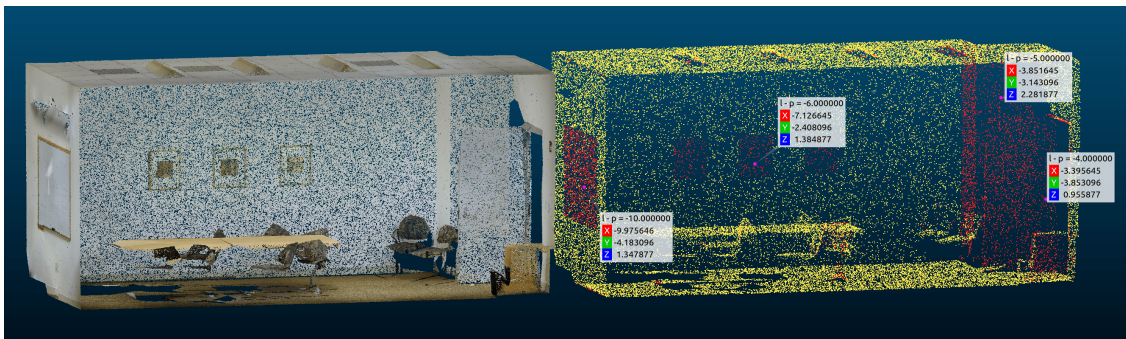


FIGURE 3.1 – Rendu d'une salle avec CloudCompare. À gauche la visualisation d'une salle brute avec les couleurs en RVB. À droite une évaluation faite par KPCConv, les points corrects sont en jaune et les erreurs en rouge.

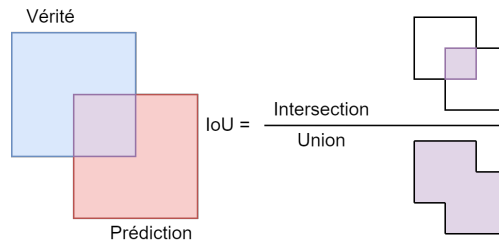


FIGURE 3.2 – Illustration du calcul de Intersection over Union

Matrice de confusion

Pour une classe, la matrice de confusion se présente sous la forme d'une matrice 2x2. Soit les colonnes donnant le label véritable (Vrai ou Faux) et les lignes le label prédit par le modèle. Étendue au nombre de classe du jeu de données, cette matrice va nous permettre de visualiser quelles classes sont les mieux reconnues mais aussi celles qui sont confondues avec d'autres.

Accuracy et mean Accuracy

L'Accuracy(Acc) ou Overall Accuracy(OA) est la proportion de points correctement labélisés par le modèle sur l'ensemble du jeu de données. Cette métrique permet d'avoir un aperçu général, cependant plus le dataset est déséquilibré, moins elle est pertinente. La mean Accuracy(mAcc) est cette fois la moyenne de l'Accuracy de chaque classe. Si le dataset n'est pas équilibré et que le modèle reconnaît moins bien les classes minoritaires alors elle sera plus impactée que l'Accuracy.

Intersection over Union et mean Intersection over Union

l'Intersection over Union(IoU) est la proportion de points correctement prédits avec un certain label x par rapport à l'union du nombre de points prédits comme x et le nombre de points dont le label est réellement x . Cette opération est illustrée par la Figure 3.2. Pouvoir visualiser l'IoU par classe au cours de l'entraînement permet de voir graphiquement quelles classes sont apprises le plus rapidement et lesquelles sont les mieux reconnues. Cependant, elle ne permet pas de faire la différence entre une classes très bien reconnue mais avec laquelle les autres classes sont confondues et une classe mal apprise mais avec laquelle peu se confondent, pour cela, il faut regarder la matrice de confusion. La mean Intersection over Union(mIoU) est la moyenne de l'IoU de chaque classe.

3.1.3 Données

Stanford large-scale 3D Indoor Spaces (S3DIS)

S3DIS[1] est un dataset dédié la compréhension de grandes scènes d'intérieur. Il existe en plusieurs formats de données 2D et 3D. Pour la partie 3D en nuages de points, il est réparti en 6 zones. La version en nuage de points comporte 695,878,620 points. Les points sont également colorés et sont répartis en 13 labels : ceiling, floor, wall, beam, column, window, door, table, chair, sofa, bookcase, board et clutter. C'est l'un des ensembles de données les plus utilisés pour comparer les méthodes sur la segmentation sémantique. Une visualisation du dataset est disponible en Figure 3.3.

Le torch-points3d propose plusieurs configurations pour le dataset S3DIS, cependant dans mon cas, seule la version 1x1 fonctionne. Dans cette version, les samples sont des colonnes de 1m x 1m et elles sont centrées sur l'origine. Ce n'est alors pas pratique pour visualiser une area entière ou tester avec une autre taille de colonne. Pour aller plus loin, il a fallu refaire les versions à partir du dataset brut.

Regrouper le dataset par salle

Initialement, le dataset est organisé de la façon suivante : il y a un répertoire par area qui lui même contient un répertoire pour chaque salle, et chaque salle contient pour chaque objet un nuage de points sous la forme d'un fichier texte. Pour regrouper le dataset par salle, il faut regrouper tous les nuages de points relatifs à une même salle tout en ajoutant une colonne représentant le label (récupéré à partir du titre du fichier).

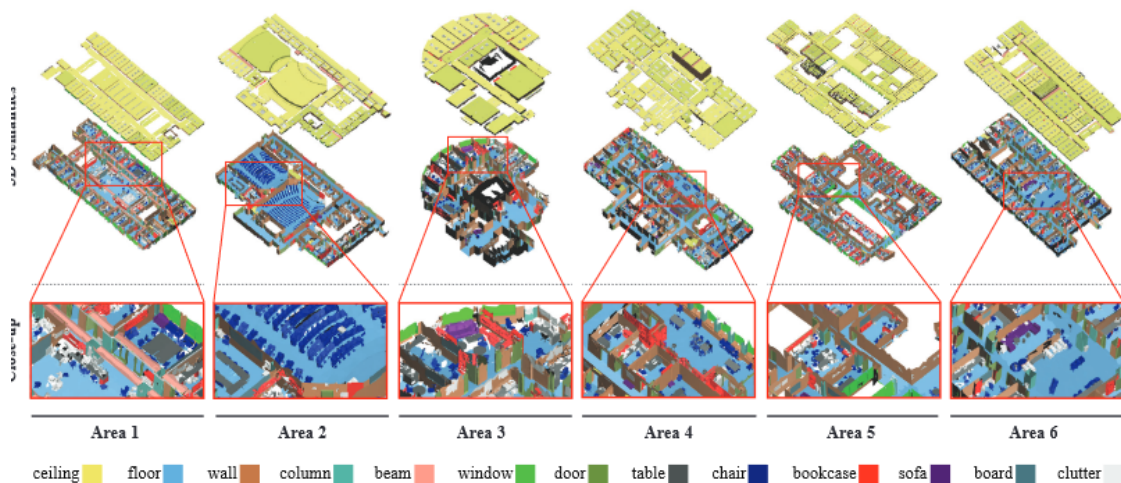


FIGURE 3.3 – Visualisation des areas de s3dis[1]

Segmenter les salles en colonnes

À partir d'un nuage on veut pouvoir y dessiner une grille, chaque case représentant une colonne de points. L'unité de la grille sera le côté des colonnes et doit être paramétrable. Le processus est décrit par Algorithm 1. Faire le quotient d'une coordonnée avec la taille des colonnes permet d'obtenir une coordonnée entière dans un repère où l'unité est la taille des colonnes. À partir des extremums en X et Y du nuage de points, on calcule la taille de la grille de colonnes selon ce repère. À cause du grand nombre de points par nuage, il est trop coûteux en calcul d'utiliser des tableaux dynamiques pour représenter les colonnes. Faire un tableau statique de la taille du nuage pour chaque colonne serait trop coûteux en mémoire vive. Il faut alors faire une première itération pour compter le nombre de points de chaque colonne, créer ces colonnes avec des tableaux statiques et enfin ajouter chaque point à la colonne correspondante.

Algorithm 1 Passer d'un nuage "pièce" à des colonnes

```
data = loadPointCloud(roompath)
minX; maxX; minY; maxY = extremumsXY(data) == columnSize
h = maxX - minX + 1
w = maxY - minY + 1
countCols = matZero(h; w)
for i in range(0, nbPoints) do
    idxX = data[i; 0] == columnSize - minX
    idxY = data[i; 1] == columnSize - minY
    countCols[idxY][idxX] + = 1
end for
cols = [[matZero(countCols[x][y]; 8) for x in range(w)] for y in range(h)]
for i in range(0, nbPoints) do
    idxX = data[i; 0] == columnSize - minX
    idxY = data[i; 1] == columnSize - minY
    countCols[idxY][idxX] = 1
    idxToAdd = countCols[idxY][idxX]
    cols[idxY][idxX][idxToAdd] = data[i]
end for
for col in Columns do
    if colSize > 100 then
        saveColumn(col; outPath=roomXY)
    end if
end for
```

Fichier de configuration et dataloader

Pour qu'un dataset personnalisé soit reconnu dans torch-points3d, il faut définir un dataloader et un fichier de configuration.

Le fichier de configuration précise le dataloader à utiliser et les transformations à appliquer au dataset pendant le preprocessing. Il est possible d'ajouter du bruit, de la symétrie, de la rotation, ou de supprimer les features d'un point de manière aléatoire mais aussi de fixer le nombre de point. Pour évaluer un modèle au mieux, la configuration définit séparément quelles transformations appliquer aux jeux de test, d'entraînement et de validation.

Le dataloader est la classe qui implémente les fonctions utiles à la manipulation du dataset. La fonction principale est "process". Elle permet de passer d'un dataset brut à un dataset reconnu par le framework. Les données sont chargées, puis transformées comme défini dans le fichier de configuration et enfin elles sont sauvegardées au format torch.

3.2 Travaux effectués

Le stage s'est déroulé d'une façon bien différente de celle que j'avais imaginé avant de le débiter. L'installation de l'environnement et la préparation des données ont demandé plus de travail que prévu. Cette partie présente les différentes tâches que j'ai effectué pendant le stage. Un récapitulatif avec une échelle de temps est visible en Figure 3.4.

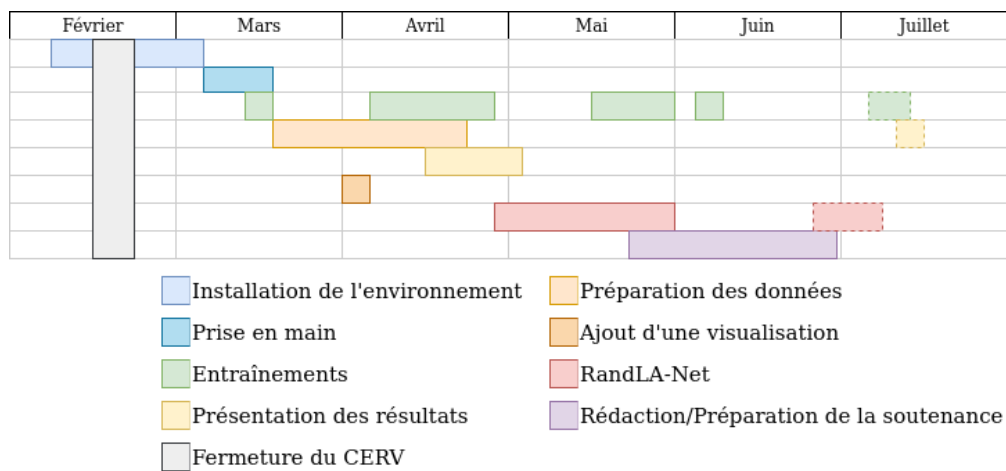


FIGURE 3.4 – Répartition des tâches au cours du stage.

3.2.1 Installation de l'environnement de travail

Le stage a débuté par l'installation des différents outils nécessaires. La mise en place fut compliquée en raison de la compatibilité (ou non) des bibliothèques entre elles et avec le système d'exploitation. Cette première étape que je pensais régler dans les premiers jours m'a pris presque trois semaines au final, les détails sont présentés dans la Section 3.3.1

3.2.2 Prise en main de torch-points3d

Pour prendre en main l'outil, j'ai commencé par suivre le tutoriel présenté dans la documentation. Puis avec les conseils de Romain, j'ai pu tester le framework sur S3DIS, un dataset plus pertinent pour le sujet du stage que celui du tutoriel.

3.2.3 Préparation des données

Le framework propose plusieurs configurations pour le dataset S3DIS, cependant dans mon cas seule la version 1x1 fonctionne. J'ai donc recréé et intégré à torch-points3d manuellement 4 versions de S3DIS dans lesquelles les échantillons sont : Des salles, des colonnes de 1m x 1m et deux avec des colonnes de 2m x 2m.

3.2.4 Analyse et présentation des résultats

Pour présenter les résultats des expérimentations et mon travail à mon encadrant, j'ai réalisé un compte rendu ainsi qu'une présentation d'une dizaine de minutes. Le Chapitre 4 présente les résultats et les analyses des performances de KPConv sur plusieurs versions de S3DIS.

3.2.5 Ajout d'une visualisation supplémentaire

Weights and Biases propose plusieurs visualisations intéressantes des performances du modèle au cours de l'entraînement. Parmi elles l'Acc, la mAcc et la mIoU, que ce soit sur le jeu d'entraînement ou le jeu de validation. Ces métriques d'évaluation sont décrites dans la section 3.1.2. Cependant, visualiser l'évolution de l'IoU pour chaque classe au cours de l'entraînement permet de mieux comprendre le fonctionnement du modèle : quelles classes sont reconnues en premières ? Lesquelles peinent à être reconnues ?

L'IoU par classe est visible dans les traces de l'entraînement. Dans un script python on analyse alors le fichier log pour récupérer les valeurs pour chaque classe à chaque epoch. On trace ensuite les courbes et enregistre la figure grâce à matplotlib. La Figure 4.3 a été produite par ce script.

3.3 Les apports du stage

Ces mois de stage ne se sont pas déroulés sans obstacles et se sont surtout ces derniers qui m'ont permis de progresser et d'améliorer mes compétences.

3.3.1 Difficultés rencontrées

Installation

Torch-points3d se base sur de nombreuses bibliothèques qui ne sont pas toutes compatibles avec les versions plus récentes que d'autres. La machine de travail était sous Windows 11 et j'avais pour consigne d'éviter de modifier le système d'exploitation. J'ai donc persévéré deux semaines à essayer d'installer les dépendances sur Windows 11 comme sur Ubuntu 18.04 via WSL2 et VirtualBox mais sans succès, notamment pour la compatibilité en CUDA et PyTorch. J'ai ensuite demandé et obtenu l'accord pour faire un dual-boot et j'ai pu installer Ubuntu 18.04 nativement. Encore quelques problèmes sont survenus à cause de l'ordre d'installation des dépendances mais au bout d'environ 3 semaines l'environnement était en place.

Torch-points3d

Torch-points3d est le seul framework qui permet de travailler avec différentes architectures de deep learning pour les nuages de points. Il a été d'une grande aide pendant ce stage. Cependant, les tutoriels et la faible documentation ne sont pas toujours à jour. Cela a rendu particulièrement difficile la prise en main de framework. D'autant plus que je n'ai pas réussi à utiliser certaines fonctionnalités sans les modifier ou les refaire, comme les différentes versions de S3DIS. De plus, il était impossible de reprendre un entraînement déjà terminé. Le problème venait du GradScaler, un objet permettant d'éviter l'underflow des poids, qui n'était pas sauvegardé et n'était pas recréé au moment de la reprise du point de contrôle. J'ai donc ajouté une ligne pour en recréer un au moment du chargement du checkpoint.

RandLA-Net

Sur la machine de travail, KPConv ne permet pas d'avoir une taille de batch supérieure à 1 avec 65000 points sans saturer la mémoire du processeur graphique. L'objectif était alors d'utiliser une version moins coûteuse en ressources de calcul : RandLA-Net. La version déjà implémentée dans torch-points3d ne fonctionnait pas. J'ai examiné le code à l'aide du debugger pour trouver le problème : Les kNNs étaient calculés sur l'ensemble des points

avant le subsampling, alors au moment d'accéder à certains points, ils n'existaient plus. Maintenant les kNNs sont calculés après le downsampling. Après avoir réussi à lancer un entraînement, je me suis rendu compte que cette implémentation utilise plus de mémoire que KPConv. J'ai tenté d'intégrer à torch-points3d une version de RandLA-Net trouvée sur github cependant les résultats sont très mauvais. Donc l'objectif de la fin de mon stage sera de tester le code sous TensorFlow présenté par [5] avec les différentes versions de S3DIS voir même de l'adapter sous PyTorch pour l'intégrer à torch-points3d.

3.3.2 Compétences

Toutes ces difficultés m'ont permis de développer mes compétences. Maintenant, grâce à la correction de RandLA-Net et l'étude bibliographique, je comprends beaucoup mieux le fonctionnement des architectures de segmentation sémantique pour les nuages de points. Les différents problèmes de torch-points3d m'ont permis de développer mon aisance avec Python et le debugger de Visual Studio Code. Sans oublier mon utilisation de LaTeX qui s'est améliorée avec la présentation des résultats et l'écriture de ce rapport.

Chapitre 4

Expérimentations

L'objectif de ces expérimentations est de voir l'impact du contexte sur la segmentation sémantique d'un nuage de points. Pour cela on va restreindre la propagation du contexte en entraînant une architecture avec des nuages de tailles différentes. Dans notre cas, c'est l'architecture KPConvDeformable qui sera utilisée car elle est déjà intégrée dans torch-points3d.

4.1 Données

Le réseau sera entraîné sur différentes versions d'un même dataset de base : S3DIS. Chaque point de ces versions est représenté par sa position (xyz) et sa couleur (rgb). La première version se nomme "room". Ici chaque sample représente une salle entière et son nombre de points est fixé à 65556. Les trois autres sont déclinées de room en la découpant en colonnes de points de 1 et 2 mètre de côté. Ce sont les versions "1x1", "2x2_8k" et "2x2_16k" et sont fixées respectivement à 4096, 8192 et 16384 points.

4.2 Implémentation et Matériel

Les entraînements ont été faits avec PyTorch 1.8.1, CUDA 11.4 et torch-points3d 1.3.0 sur une carte NVidia Quadro RTX 3000 (portable). La carte graphique moyenne gamme ne possède que 6Go de mémoire, cela va causer des limitations dans le nombre de points que l'on peut traiter. Pour équilibrer la différence de taille entre les versions du dataset, on va garder une même proportion de points par rapport à la taille de batch. Les batch sizes des versions room, 2x2_16k, 2x2_8k et 1x1 seront donc respectivement 1, 4, 8 et 16. Les entraînements ont duré 300 epochs.

4.3 Résultats

Les résultats des tests sont représentés par différentes tables, courbes et figures.

ID	Classe	IoU 1x1	IoU 2x2 8k	IoU 2x2 16k	IoU room
0	Beam	0.10	1.49	0.82	0.01
1	Board	2.98	4.00	0.72	0.00
2	Bookcase	42.02	48.66	44.81	44.72
3	Ceiling	91.05	89.70	90.23	87.75
4	Chair	65.61	64.67	61.59	62.43
5	Clutter	38.14	35.19	32.70	36.39
6	Column	4.28	0.47	0.92	4.17
7	Door	27.63	16.77	8.89	10.61
8	Floor	95.28	97.03	97.09	97.81
9	Sofa	14.59	22.29	7.73	14.10
10	Table	63.40	66.18	64.17	67.62
11	Wall	66.76	70.86	69.36	64.70
12	Window	6.15	23.30	20.58	8.22

TABLE 4.1 – Labels et classes de S3DIS, ainsi que l’IoU par classe de chaque version à la fin de l’entraînement.

4.3.1 Nombre de points et taille de batch

Les courbes en Figures 4.1 à 4.3 permettent de se rendre compte de l’importance du batch size. À cause de son entraînement avec une taille de batch de 1, la version room converge rapidement mais fait des variations très amples même après 100 epochs. Sur la Table 4.2, on peut voir l’accuracy, la mean Accuracy et la mIoU des quatre versions. Les versions 2x2 ont les mêmes colonnes de points à la différence que 2x2_8k possède une densité de points 2 fois moins importante que 2x2_16k mais avec une taille de batch deux fois supérieure. Les Tables 4.1 et 4.2 montrent qu’à l’exception des classes ”floor” et ”ceiling”, la version avec la plus grande taille de batch surpasse l’autre.

Sur la Figure 4.3, on voit distinctement que le réseau apprend les classes les plus représentées ou plus faciles en premier et ensuite affine les prédictions pour les suivantes. On pourrait

	Acc	mAcc	mIoU
1x1	79.30	47.66	39.84
2x2 8k	79.32	50.81	41.58
2x2 16k	78.37	45.61	38.44
room	76.53	44.85	38.35

TABLE 4.2 – Accuracy, mean Accuracy et mean Intersection over Union de chaque version à la fin de l’entraînement.

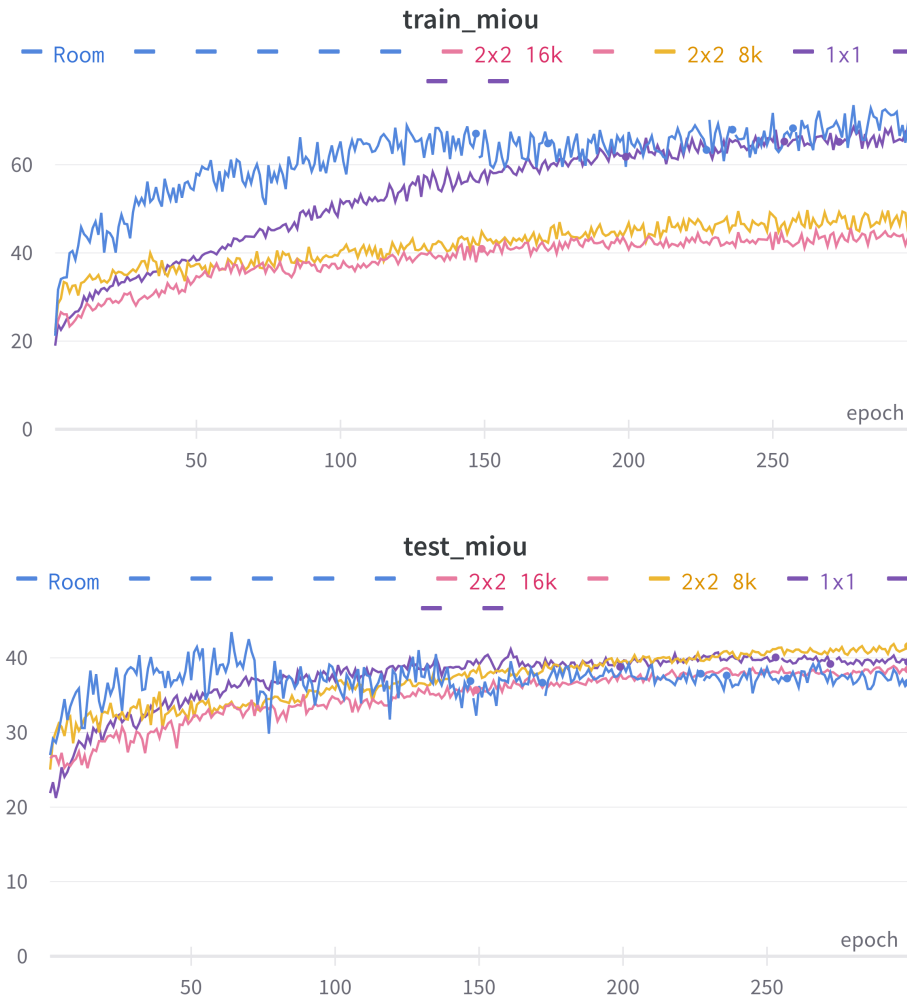


FIGURE 4.1 – Évolution de la mIoU sur le jeu d’entraînement et sur le jeu de test au cours de l’entraînement. Les versions 1x1, 2x2 8k, 2x2 16k et room sont représentées respectivement en violet, orange, rose et bleu.

sembler que travailler à l’échelle d’une salle entière permettrait d’avoir une meilleure vision d’ensemble qu’avec des colonnes de 1x1. En comparant les courbes sur la Figure 4.1, on voit que la version room apprend mieux le jeu d’entraînement mais elle a du mal à généraliser. Le nombre de sample joue aussi sur la capacité d’apprentissage : de l’ordre de 200 pour room et 6500 pour 1x1.

4.3.2 Différences d’apprentissage

Sur la matrice de confusion en Figure 4.4 peut voir une diagonale se dessiner. Lorsqu’une classe n’est pas apprise, le réseau la classe comme "clutter" (5) ou "wall" (11) par défaut. Justement les éléments de la classe 5 sont mieux reconnus par la version room mais les autres classes sont aussi plus confondues avec celle-ci. La Table 4.1 montre que la version

1x1 domine sur les classes "ceiling", "chair", "clutter", "column" et "door". Hors-mis le plafond, ce sont toutes des entités dont la base se rapproche plus du mètre. On peut faire le même constat pour les autres versions : la version 2x2 est la plus performante sur les classes "beam", "board", "bookcase", "sofa" et "window", ces éléments ont aussi une taille qui correspond à la version 2m x 2m. Cependant la version room n'est meilleure qu'avec les classes "table" et "floor". La table pouvant être supérieure à 2m comme inférieure. En généralisant, on voit que le modèle reconnaît mieux les objets qui sont à son échelle.

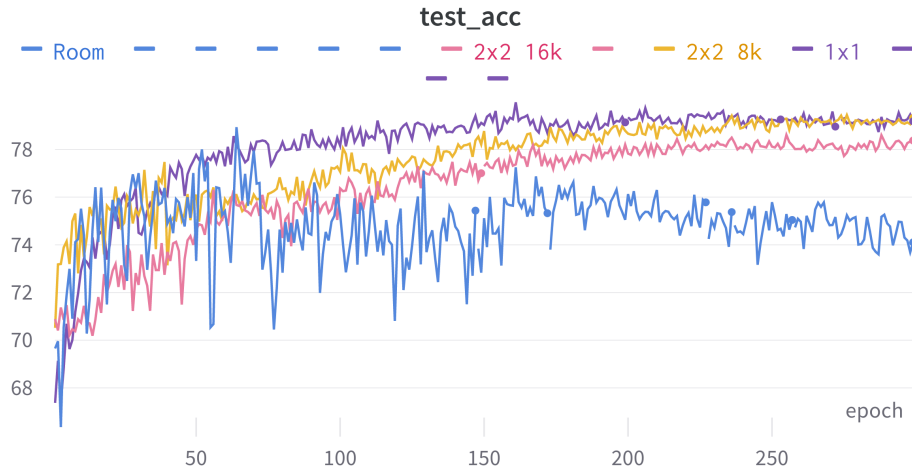


FIGURE 4.2 – Évolution de l'accuracy sur le jeu de test au cours de l'entraînement. Les versions 1x1, 2x2 8k, 2x2 16k et room sont représentées respectivement en violet, orange, rose et bleu.

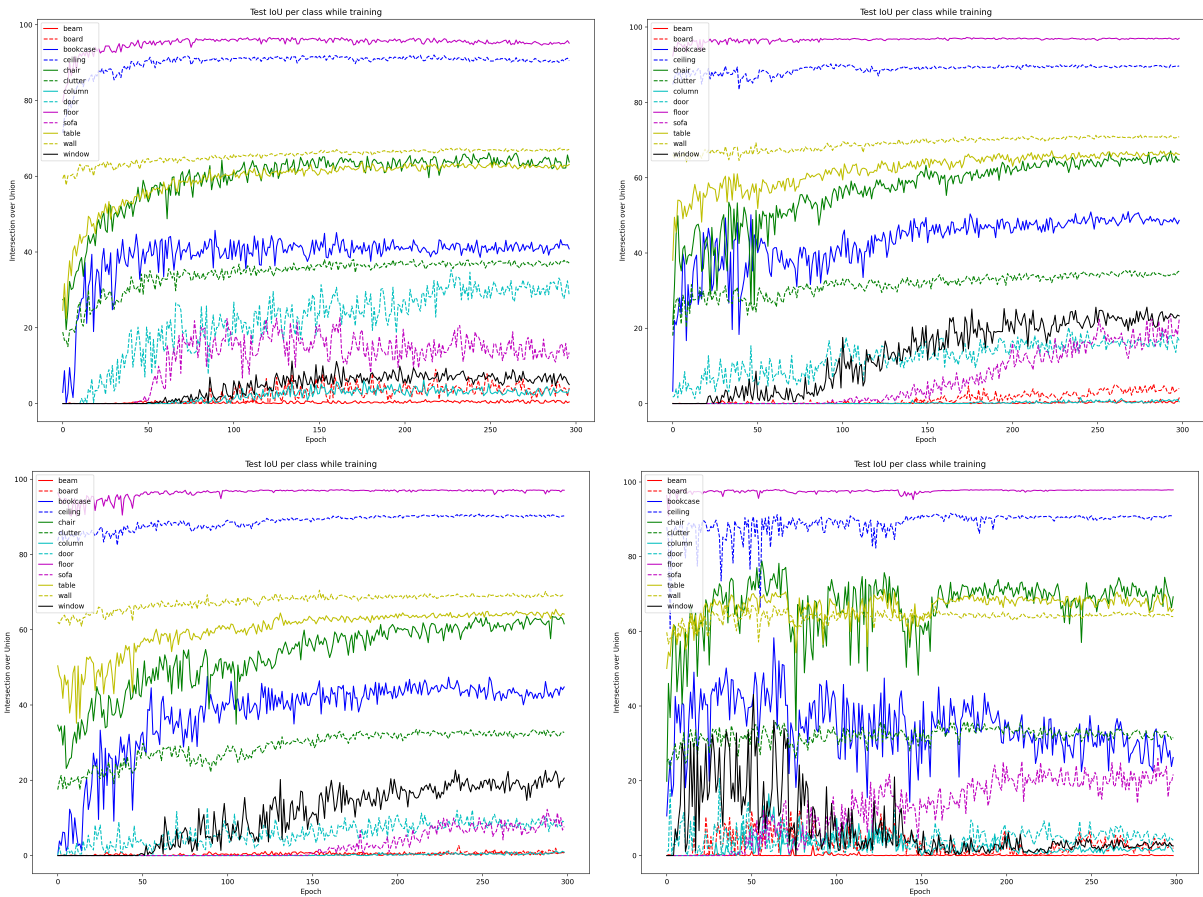


FIGURE 4.3 – Évolution de l’IoU par classe au cours de l’entraînement de la version 1x1 (en haut à gauche), 2x2 8k(en haut à droite), 2x2 16k (en bas à gauche) et room (en bas à droite).

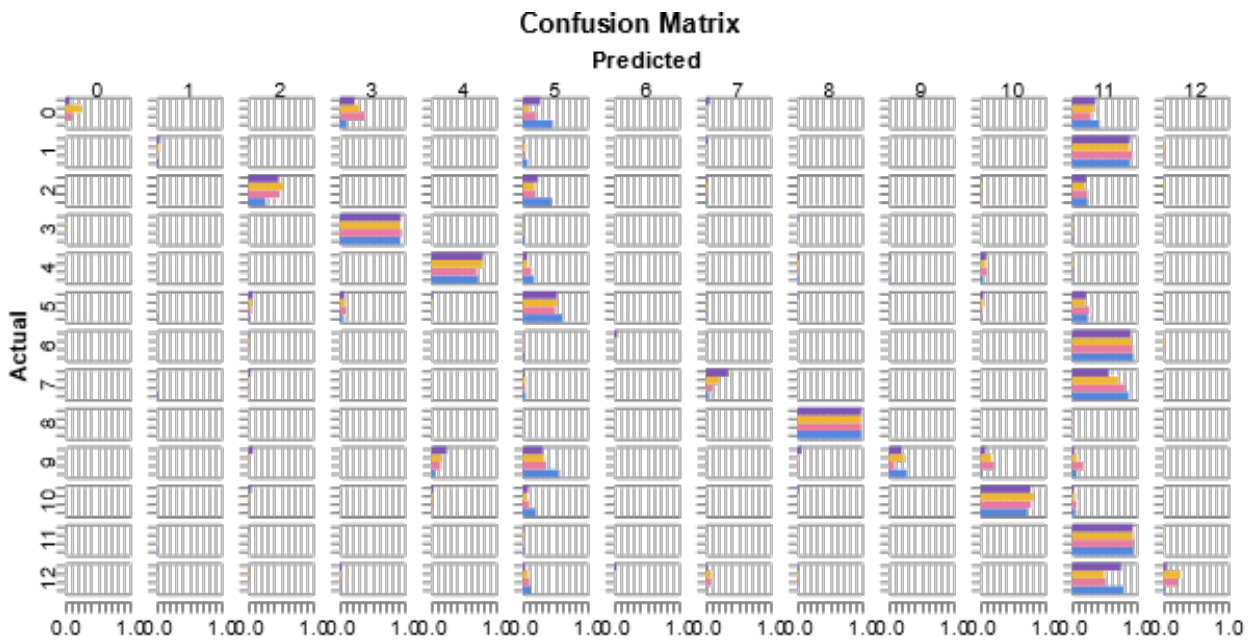


FIGURE 4.4 – Matrice de confusion produite sur le jeu de test. Les versions 1x1, 2x2 8k, 2x2 16k et room sont représentées respectivement en violet, orange, rose et bleu.

Chapitre 5

Conclusion

L'absence d'ordre et de structure dans les nuages de points ajoute une difficulté par rapport au traitement d'images. Ce n'est que depuis 2017 avec la publication de PointNet[10] que l'on sait faire des réseaux de neurones qui travaillent directement sur les nuages de points et non une projection. Ces architectures sont en grande majorité de type Encoder-Decoder. Elles se différencient en elles principalement par leur méthode pour encoder l'information mais aussi par le format des données d'entraînement. Pour le dataset S3DIS, les méthodes utilisent comme samples des colonnes de 1m x 1m, des salles entières, ou encore des sphères. Le choix du format est souvent dirigé par les ressources matérielles mais il n'y a pas d'étude comparative sur les avantages et inconvénients des formats. L'objectif est de comprendre l'impact de l'échelle des échantillons de données sur la segmentation sémantique d'un nuage de points.

À partir de S3DIS, on a pu créer des samples de trois tailles : une salle entière, une colonne de 1m x 1m et une colonne de 2m x 2m. La version 2m x 2m est déclinée avec deux densités de points. Entraîner un réseau KPConvDeformable[13] avec ces quatre versions a permis de comparer les différences d'apprentissage en fonction de la taille de l'échantillon.

Les résultats montrent qu'utiliser le nuage de points dans sa totalité permet d'avoir le maximum d'informations mais cela demande plus de ressources de calcul pour pouvoir l'entraîner correctement. L'information peut se diluer pendant de downsampling et ainsi les petits objets sont moins bien reconnus. En segmentant le nuage d'origine en colonnes, on améliore les performances sur les entités de la même échelle que la colonne. Cependant, si des entités sont plus grandes et découpées, cela empêche au réseau de bien en apprendre les motifs et baisse les performances sur ces classes.

Un batch size égal à 1 pour l'entraînement avec la version room cause une convergence instable. Il serait intéressant d'utiliser une méthode moins coûteuse en ressources ou une carte graphique plus puissante pour pouvoir tester avec une taille de batch de 3 ou 4.

Chapitre 6

Retour d'expérience

Que ce soit au niveau personnel ou professionnel, ce stage de recherche aura été une expérience enrichissante.

L'état de l'art et les fonctionnalités hors-service de torch-points3d m'ont permis de beaucoup apprendre sur les réseaux de neurones artificiels et l'apprentissage profond. J'ai aussi pris l'habitude d'utiliser le debugger, ce qui est un réel gain de temps.

Pendant ce stage, programmer m'a manqué. En ajoutant l'aspect d'exploration de la recherche avec lequel j'ai eu un peu de mal, je vais orienter mon projet professionnel vers un poste d'ingénierie en industrie plutôt que vers un doctorat, mais, je n'exclus pas cette possibilité à moyen terme.

Après ces 5 mois passés au Centre Européen de Réalité Virtuelle, je souhaite avoir une expérience professionnelle en industrie et peut être dans un autre domaine pour ne pas m'enfermer dans l'apprentissage automatique, je pense notamment aux systèmes embarqués.

Bibliographie

- [1] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. page 10, 2016.
- [2] Alexandre Boulch, Joris Guerry, Bertrand Le Saux, and Nicolas Audebert. SnapNet : 3d point cloud semantic labeling with 2d deep segmentation networks. 71 :189–198, 2017. Publisher : Elsevier.
- [3] Thomas Chaton, Chaulet Nicolas, Sofiane Horache, and Loic Landrieu. Torch-points3d : A modular multi-task framework for reproducible deep learning on 3d point clouds. In *2020 International Conference on 3D Vision (3DV)*. IEEE, 2020.
- [4] Siqi Fan, Qiulei Dong, Fenghua Zhu, Yisheng Lv, Peijun Ye, and Fei-Yue Wang. SCF-net : Learning spatial contextual features for large-scale point cloud segmentation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14499–14508, 2021. ISSN : 2575-7075.
- [5] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. RandLA-net : Efficient semantic segmentation of large-scale point clouds. (arXiv :1911.11236), 2020. type : article.
- [6] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise convolutional neural networks. Number : arXiv :1712.05245.
- [7] Mingyang Jiang, Yiran Wu, Tianqi Zhao, Zelin Zhao, and Cewu Lu. PointSIFT : A SIFT-like network module for 3d point cloud semantic segmentation. Number : arXiv :1807.00652.
- [8] Loic Landrieu and Mohamed Boussaha. Point cloud oversegmentation with graph-structured deep metric learning. Number : arXiv :1904.02113.
- [9] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. Number : arXiv :1711.09869.

- [10] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet : Deep learning on point sets for 3d classification and segmentation. (arXiv :1612.00593), 2017. type : article.
- [11] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++ : Deep hierarchical feature learning on point sets in a metric space. (arXiv :1706.02413), 2017. type : article.
- [12] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. OctNet : Learning deep 3d representations at high resolutions. Number : arXiv :1611.05009.
- [13] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. KPConv : Flexible and deformable convolution for point clouds. Number : arXiv :1904.08889.
- [14] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. Number : arXiv :1801.07829.
- [15] Xiaoqing Ye, Jiamao Li, Hexiao Huang, Liang Du, and Xiaolin Zhang. 3d recurrent neural networks with context fusion for point cloud semantic segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.
- [16] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. PointWeb : Enhancing local neighborhood features for point cloud processing. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5560–5568. IEEE, 2019.
- [17] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ADE20k dataset. Number : arXiv :1608.05442.