Apprentissage par réseaux de neurones profonds sur les nuages de points 3D

William Guimont-Martin
Département d'informatique et de génie logiciel
Université Laval
Québec, Canada
william.guimont-martin.1@ulaval.ca

Résumé

Les nuages de points 3D sont des données difficiles à traiter pour les réseaux de neurones puisqu'il s'agit de données non structurées, non ordonnées et à densité non homogène. Ainsi, afin de résoudre des tâches reliées aux nuages de points avec l'apprentissage profond, il est nécessaire d'utiliser des techniques aptes à gérer les particularités des nuages de points. Ces approches se divisent en deux grandes catégories : les approches structurées et les approches non structurées. Cet article a pour but de présenter les principales approches pour traiter des nuages de points par apprentissage profonds, de les comparer et d'en fournir une synthèse.

1 Introduction

Contrairement aux caméras qui projettent le monde réel tridimensionnel sur un plan 2D pour générer des images, perdant ainsi l'information sur la profondeur des objets, les capteurs 3D permettent de mesurer directement le monde en trois dimensions. Les capteurs 3D, comme les lidars, permettent d'obtenir un nuage de points 3D représentant la scène mesurée. Les nuages de points, qui représentent la sortie de la plupart des capteurs 3D [Wang et al., 2018], sont des ensembles de coordonnées 3D. Le traitement de ces données par apprentissage profond est un problème important, puisqu'il ouvre la porte à plusieurs applications intéressantes, notamment la détection de véhicules pour les voitures autonomes ([Lang et al., 2018], [Yan et al., 2018]), la compréhension sémantique des scènes 3D ([Nunes et al., 2022], [Xie et al., 2020]), et la génération de modèles de conception assistée par ordinateur (CAO) à partir de scans 3D d'objets ([Lê et al., 2021]).

Malgré les nombreuses applications possibles de l'apprentissage par réseaux de neurones profonds sur des nuages de points, ce type de données est particulièrement difficile à traiter pour ceux-ci. Contrairement aux images, dont la structure en grille de pixels les rend faciles à traiter par les réseaux de neurones convolutifs (CNN), les nuages de points présentent des particularités les rendant plus difficiles à gérer. Bello et al. [2020] résume ces particularités en trois catégories : l'irrégularité des données, le manque de structure et le fait que les points dans un nuage de points ne sont pas ordonnés.

Premièrement, la répartition des points dans l'espace est irrégulière : certaines zones du nuage de points possèdent une forte densité de points alors que d'autres seront presque vides. Ceci est notamment le cas pour les scans lidars, où les régions près du capteur sont généralement plus denses que les régions éloignées.

Deuxièmement, les nuages de points ne possèdent pas de structure ; les points ne sont pas disposés en une grille régulière comme le sont les pixels d'une image. Plutôt, les points sont répartis dans l'espace, et la distance entre un point et ses voisins n'est pas constante. Ce manque de structure rend impossible l'utilisation de réseaux CNN sur la liste des points [Bello et al., 2020].

Troisièmement, les points dans un nuage de points ne sont pas ordonnés puisque les nuages de points sont des ensembles. Ainsi, l'ordre dans lequel on présente les points au modèle ne devrait pas changer sa sortie. Ceci demande aux modèles d'être invariant à la permutation des points.

Afin de pouvoir traiter convenablement les nuages de points à l'aide d'apprentissage profond, il est nécessaire d'utiliser des techniques prenant en compte les particularités des nuages de points. Cet article présente, par le biais d'articles clés exemplifiant les concepts en jeux, les principales techniques utilisées dans la littérature. Les articles étudiés sont les suivants :

— PointPillars : Fast Encoders for Object Detection From Point Clouds [Lang et al., 2018] ;
— FKAConv : Feature-Kernel Alignment for Point Cloud Convolution [Boulch et al., 2020] ;
— PointNet++ : Deep Hierarchical Feature Learning on Point Sets in a Metric Space ; [Qi et al., 2017] ;
— DGCNN : Dynamic Graph CNN for Learning on Point Clouds [Wang et al., 2018].

La Section 2 présente une taxonomie des différentes approches de traitement des nuages de points pour l'apprentissage profond. Ensuite, la6 Section 3 et la Section 4 proposent, respectivement, une comparaison et une synthèse des approches présentées dans les articles à l'étude.

2 Apprentissage par réseaux de neurones profonds sur des nuages de points

Inspiré par la catégorisation de Bello et al. [2020], on propose une taxonomie des méthodes de traitement des nuages de points dans la Figure 1. Pour soucis de complétude, on décrit également des techniques qui ne sont pas utilisées dans les quatre articles discutés. Ces techniques se séparent en deux grandes catégories selon leur façon de gérer le manque de structure dans les nuages de points : les méthodes structurées et les méthodes non structurées.

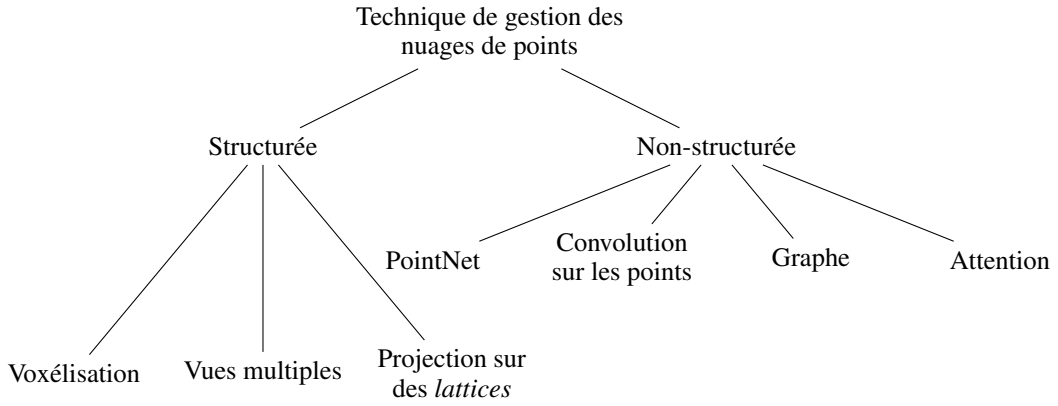

FIGURE 1 – Taxonomie des méthodes de traitement des nuages de points en apprentissage profonds

2.1 Approches structurées

Les approches structurées imposent une structure aux nuages de points afin de les rendre plus faciles à traiter par les réseaux de neurones. Cette structure peut prendre plusieurs formes : une grille de voxels, la transformation du nuage de points en une séquence d'images et la projection des points sur des *lattices*.

2.1.1 Voxélisation

La voxélisation consiste à transformer le nuage de points en une grille 3D régulière d'éléments de volume 3D, aussi connus sous le nom de voxels. Dans le cas de VoxNet [Maturana and Scherer, 2015], la voxélisation prend la forme d'une grille d'occupation où chaque voxel contient la densité de points à l'intérieur de son volume. Cette représentation en grille 3D permet d'utiliser des opérations de convolution 3D.

Dans le cas de scans lidars capturés par de véhicules en milieux urbains, comme dans le jeu de données KITTI [Geiger et al., 2013], il est possible d'utiliser une voxélisation à vue de vol d'oiseau (*bird's eye view*, ou *BEV*) [Yang et al., 2018]. Ce type de voxélisation consister à visualiser le nuage de points vus d'en haut (à vol d'oiseau), et de séparer le nuage de points en une grille 2D de voxels au sol. Cette approche donne son nom à PointPillars [Lang et al., 2018], puisque chaque voxel contient un volume vertical délimité par la grille 2D au sol, leur donnant la forme de piliers. Afin de résumer la géométrie des points contenus à l'intérieur de chaque voxel, PointPillars utilise une version simplifiée de PointNet [Qi et al., 2016] sur les points contenus par ceux-ci. Ceci a pour effet d'encoder le nuage de points en une pseudo-image sur laquelle il est possible d'utiliser la convolution 2D [Yang et al., 2018], permettant ainsi de réutiliser des architectures de vision par ordinateur.

2.1.2 Vues multiples

Dans le même ordre d'idées, il est possible de réduire le problème du traitement de nuages de points en un problème de vision par ordinateur en générant une séquence d'images du nuage de points. En effet, les méthodes à vues multiples utilisent des caméras virtuelles afin de générer des images du nuage de points à partir de plusieurs points de vue. Ensuite, il est possible d'utiliser directement des modèles de vision par ordinateur afin de résoudre la tâche en jeu.

Cette approche est notamment utilisée par You et al. [2018] qui génère des images tout autour du modèle de l'objet 3D ou le nuage de points afin de le classifier à l'aide de réseaux CNN. L'information des différentes images est fusionnée grâce à une opération de *pooling*.

2.1.3 Projection sur des *lattices*

Afin de contrevenir au manque de structure des nuages de points, il est possible de projeter et d'interpoler les points sur une grille régulière de points (une *lattices*) sur laquelle il est possible de définir une opération de convolution. Cette projection sur une grille, similaire à la voxélisation, permet d'imposer une structure au nuage de points et l'utilisation de la convolution.

Ce type de méthode est notamment utilisé par SplatNet [Su et al., 2018], qui projette les points sur une *lattices* à 6 dimensions, et SFCNN [Rao et al., 2019], qui utilise une *lattices* sphérique.

2.2 Approches non structurées

Alors que les méthodes structurées imposent une structure aux nuages de points, les méthodes non structurées visent à plutôt traiter directement les nuages de points. Il existe plusieurs types d'approches ne nécessitant pas de structurer les nuages de points : les méthodes basées sur PointNet, la convolution sur les points, les réseaux sur les graphes et les mécanismes d'attention.

2.2.1 PointNet

Les méthodes basées sur PointNet [Qi et al., 2016] tirent avantage des fonctions symétriques afin d'être invariantes à l'ordre dans lequel les points sont présentés au réseau. Plus particulièrement, PointNet projette chaque point séparément à l'aide d'une couche pleinement connectée (*fully connected layer*) et applique la fonction symétrique de *maxpooling* afin de résumer le nuage de points en un seul vecteur.

Aussi, pour rester invariant aux transformations géométriques du nuage de points (rotation et translation de tous les points du nuage), PointNet utilise un sous-réseau permettant d'appliquer une transformation affine à tous les points.

PointNet++ [Qi et al., 2017] ajoute à PointNet une notion de hiérarchie permettant de tenir en compte des détails locaux des nuages de points à différentes échelles. PointNet++ consiste à appliquer PointNet sur le voisinage de points choisis par l'algorithme *farthest point sampling* (FPS). Le voisinage d'un point choisis par FPS peut être déterminé par l'algorithme des K plus proches voisins (KNN) ou par l'ensemble des points contenus dans une balle de rayon r autour du point (requête en balle ou *ball query*).

2.2.2 Convolution sur les points

Alors que les méthodes structurées tentaient de structurer les données afin de pouvoir y appliquer la convolution, la méthode de convolution sur les points propose plutôt d'adapter le concept de la convolution à des domaines sans structure. FKAConv [Boulch et al., 2020] généralise la notion de convolution en y ajoutant une projection permettant d'adapter l'espace des points à celui du *kernel*. Cette projection permet d'aligner les points sur un *kernel* et d'ainsi réaliser une convolution. La projection est déterminée à partir des K plus proches voisins autour du point sur lequel on applique la convolution.

Comme le voisinage est déterminé par les K plus proches voisins, la distance entre le point central et son voisin le plus éloigné peut varier grandement dans les zones moins denses. Pour contrer cet effet, FKAConv applique une normalisation du voisinage.

Comme la convolution sur chacun des points du nuage demande beaucoup de temps de calcul, il est possible d'accélérer le temps d'entraînement en ne réalisant des convolutions que sur un sous-ensemble des points. FKAConv propose l'algorithme *Efficient point sampling with space quantization* (*EPSSQ*) afin d'efficacement échantillonner ces points.

2.2.3 Graphe

Les méthodes basées sur les graphes, comme DGCNN [Wang et al., 2018], transforment les nuages de points en graphes sur lesquels il est possible de réaliser une convolution sur les arêtes. DGCNN construit un graphe en connectant avec des arêtes chaque point à ses K plus proches voisins. Ensuite, il est possible d'utiliser une convolution sur les arêtes afin d'extraire de l'information sur la géométrie locale autour d'un point. Similairement à la convolution sur les images, où l'opération de convolution génère une nouvelle image appelée une *feature map*, DGCNN génère un nouveau nuage de points dans un autre espace (*feature space*). Ceci permet d'appliquer la convolution sur les arêtes à nouveau, mais cette fois-ci dans l'espace abstrait des descripteurs (*feature space*).

2.2.4 Attention

Avec ses récents succès dans le domaine du traitement de la langue naturelle ([Vaswani et al., 2017], [Devlin et al., 2018]), les *transformers* sont devenus des outils polyvalents en apprentissage profond, trouvant des applications dans plusieurs domaines notamment celui de la vision par ordinateur (Carion et al. [2020], [Dosovitskiy et al., 2020]).

Inspiré par les nombreuses applications des *transformers*, Point Transformer [Zhao et al., 2021] adapte les *transformers* au monde des nuages de points. Les *transformers* sont particulièrement adaptés aux nuages de points puisqu'ils représentent une opération sur des ensembles non ordonnés de jetons (*tokens*). Similairement, Point-BERT [Yu et al., 2021] adapte les *transformers* et les concepts d'apprentissage auto supervisé (*self-supervises learning*) de BERT [Devlin et al., 2018] aux nuages de points.

3 Comparaison

Les quatre articles à l'étude ([Lang et al., 2018], [Boulch et al., 2020], [Qi et al., 2017] et [Wang et al., 2018]), malgré leur but commun d'adapter les réseaux de neurones aux nuages de points, présentent des différences importantes. Cette section a pour but de discuter les principales différences entre ces méthodes.

3.1 Traitement des nuages de points

Chacune des méthodes étudiées tente de gérer les particularités des nuages de points, mais elles s'attaquent au problème de façons différentes. Les approches structurées adaptent les données afin de pouvoir appliquer la convolution, alors que les méthodes non structurées adaptent plutôt les architectures aux nuages de points afin de pouvoir les traiter directement.

Les méthodes structurées s'attaquent directement au manque de structure des nuages de points en les représentants sous la forme d'une grille de voxels. Ceci permet d'aisément réutiliser les opérations de convolutions qui sont bien étudiées par la communauté d'apprentissage profond. Plus particulièrement,

la voxélisation à vue de vol d'oiseau de PointPillars permet de résumer le nuage de points en une pseudo-image sur laquelle il est possible d'utiliser les architectures de détection 2D couramment utilisées en vision par ordinateur. Ceci représente un avantage considérable puisqu'il est possible d'adapter de nombreuses architectures de vision par ordinateur.

Les méthodes non structurées, n'imposant pas de structure aux données, ne permettent généralement pas de réutiliser des architectures de vision par ordinateur, et demande plutôt des architectures plus spécialisées. Les trois méthodes non structurées présentées conservent l'information (les *features*) par point, ce qui demande des têtes de détections, de classification ou de détection spécifiques à cette représentation.

Malgré cet avantage des méthodes structurées, la voxélisation présente plusieurs limitations. Comme soulevé par Qi et al. [2016], la représentation en voxels d'un nuage de points demande une quantité de mémoire plus importante et cause des artefacts de quantisation nuisant aux modèles. En effet, lors du découpage du volume en grille 3D, il y a un compromis entre la quantité de mémoire utilisée et la précision de la voxélisation. Une grille fine permet d'adéquatement représenter le nuage de points, mais demande une quantité prohibitive de mémoire. Une grille grossière, malgré sa plus faible empreinte mémoire, va causer des artefacts de quantisation importants [Bello et al., 2020].

Les méthodes non structurées, quant à elles, n'ont pas à gérer ce compromis puisque les opérations sont réalisées directement sur les points. Ainsi, les modèles non structurés évitent les erreurs de quantisation et utilisent généralement moins de mémoire que la voxélisation [Qi et al., 2016].

3.2 Tâches et jeux de données

Les méthodes structurées et non structurées sont chacune plus adaptées à certaines tâches qu'à d'autres et, dans certains cas, demande un type de nuage de points particulier afin d'être efficace.

Les nuages de points capturés par un lidar sur un véhicule possèdent une structure reflétant la nature structurée de l'environnement urbain. Ceux-ci sont généralement organisés selon un plan 2D, le sol, sur lequel des objets, comme des voitures, des piétons et des cyclistes, reposent. Cette structure inhérente à ce type de nuages de points permet d'utiliser la voxélisation à vue de vol d'oiseau de PointPillars. Ainsi, PointPillars est bien adapté à des jeux de données comme KITTI [Geiger et al., 2013], KITTI-360 [Liao et al., 2021], Waymo Open Dataset [Sun et al., 2020] et nuScenes [Caesar et al., 2020] qui sont tous des jeux de données de scans lidar capturés depuis un véhicule.

Cette technique de voxélisation n'est toutefois pas adaptée aux autres types de nuages de points. C'est le cas des nuages de points générés à partir des jeux de données ModelNet40 [Wu et al., 2015] et ShapeNet [Chang et al., 2015]. Ces jeux de données contiennent des modèles 3D qu'il est possible de convertir en nuage de points en échantillonnant des points sur les surfaces des modèles [Qi et al., 2016]. Ce type de données, ne possédant pas de plan 2D principal, n'est pas adapté à la voxélisation à vue de vol d'oiseau de PointPillars. Les méthodes non structurées sont plus adaptées dans ce cas.

PointPillars a été conçu avec le but de détecter des objets en 3D dans des nuages de points. Ainsi, l'architecture n'est pas adaptée à réaliser de la classification ou de la segmentation sémantique. À l'inverse, les méthodes non structurées, c'est-à-dire PointNet++, FKAConv et DGCNN, conservent l'information (les *features*) par point, ce qui les rend particulièrement adaptés à la segmentation et la classification.

3.3 Type d'échantillonnage

Les méthodes non structurées doivent échantillonner les points autour desquels les opérations (la convolution sur les points, un PointNet ou la convolution sur les arêtes) seront réalisées. Dans le cas de PointNet, l'échantillonnage des points est fait avec l'algorithme *farthest point sampling* (FPS). FKAConv propose un algorithme similaire pour atteindre le même but : l'algorithme *Efficient point sampling with space quantization* (*EPSSQ*). DGCNN, opérant sur l'ensemble des points, n'a pas besoin d'échantillonnage. Pour ce qui est de PointPillars, les points étant déjà divisés entre les voxels, il n'est pas nécessaire d'échantillonner le nuage de points.

3.4 Type de regroupement

Une fois les points sur lesquels on applique les opérations ont été échantillonnés, il faut déterminer le voisinage de ces points. Ce voisinage est déterminé de différentes façons selon les articles.

Pour PointPillars, les points d'un même voxel sont considérés comme faisant partie d'un même voisinage ; il n'est donc pas nécessaire d'exécuter un algorithme de regroupement additionnel.

Les approches non structurées, quant à elles, utilisent soit les K plus proches voisins, ou une requête en balle (*ball query*). KNN a l'avantage d'être plus rapide qu'une requête en balle [Boulch et al., 2020], mais fait en sorte que la taille de voisinage peut grandement varier [Boulch et al., 2020]. Pour contrer cet effet, qui peut être néfaste aux capacités de généralisation des modèles [Qi et al., 2017], FKAConv utilise une normalisation de la taille du voisinage.

4 Synthèse

À partir des observations faites à la Section 3, il est possible de tisser des liens entre les différentes techniques étudiées dans cet article. La Table 1 résume les observations faites sur les quatre articles dans une matrice de synthèse.

4.1 Traitement des nuages de points

Chaque méthode présentée extrait de l'information sur la géométrie locale des nuages de points en réalisant une opération sur le voisinage des points. Ceci est très similaire aux CNN qui extraient des patrons locaux avec les couches convolutionnelles. Ce lien est présent autant dans les méthodes non structurées, où l'on applique hiérarchiquement des extracteurs de patrons locaux aux nuages de points, que dans les approches structurées qui utilisent directement la convolution. Ainsi, on peut dire que les techniques discutées dans cet article procèdent de façon similaire aux réseaux convolutifs.

On remarque aussi que chaque méthode utilise une représentation différente des nuages de points. PointPillars considère avant tout le nuage de points comme appartenant à un volume. FKAConv et PointNet++ considèrent les nuages de points comme étant des ensembles de coordonnées 3D. DGCNN représente plutôt le nuage de points comme des regroupements locaux de points fortement corrélés, et explicite la relation entre les points à l'aide d'un graphe.

4.2 Champs réceptifs

L'utilisation des algorithmes de regroupement discutés à la Section 3.4 permet de définir la notion de champ réceptif pour ces types de réseaux ([Boulch et al., 2020], [Qi et al., 2017] et [Wang et al., 2018]). Contrairement aux réseaux à convolution où les champs réceptifs locaux sont de taille fixe, pour les méthodes utilisant KNN, la taille des champs réceptifs va varier selon les données. Ceci peut s'avérer être un inconvénient puisque Qi et al. [2017] propose que des champs réceptifs de taille constante permettent une meilleure généralisation dans l'espace. Une requête en balle permet d'obtenir un champ réceptif de taille constante, mais demande plus de temps de calcul [Boulch et al., 2020]. Ainsi, la gestion des champs réceptifs est un compromis important pour l'apprentissage par réseaux de neurones profonds sur les nuages de points.

4.3 Types d'invariance

Dans la résolution de plusieurs tâches, comme la classification, les nuages de points peuvent subir plusieurs transformations sans que la sortie du modèle change. C'est notamment le cas lors de la translation et la rotation des nuages de points 3D de ModelNet [Wu et al., 2015], où ce type de transformation ne change pas la classe du nuage de points. Ainsi, les modèles ont avantage à présenter une invariance aux diverses transformations que peuvent subir les nuages de points. Parmi les approches discutées, toutes possèdent une invariance à la translation, mais seulement FKAConv et PointNet++ sont invariants aux rotations.

Similairement, comme les points dans un nuage de points ne sont pas ordonnés, les modèles doivent être invariants à la permutation des points dans leurs entrées. Changer l'ordre dans lequel on présente

les points au modèle ne devrait pas changer sa sortie. Toutes les approches présentées sont invariantes à la permutation des points.

5 Conclusion

L'apprentissage par réseaux de neurones profonds sur les nuages de points 3D présente plusieurs défis étant données les nombreuses particularités de ce type de données. En effet, les nuages de points étant des données irrégulières, sans structure et non ordonnées, sont difficiles à traiter pour les réseaux de neurones. Afin de gérer ces particularités, les approches structurées imposent une structure aux données, alors que les méthodes non structurées s'adaptent aux nuages de points afin de pouvoir les traiter directement. On a proposé une comparaison de ces différentes approches, ainsi qu'une discussion sur les liens qu'il est possible de tisser entre elles. Avec l'accessibilité croissante des capteurs 3D, comme les lidars sur les véhicules autonomes et les caméras à temps de vol, l'apprentissage profond sur des nuages de points risque de prendre beaucoup d'essor grâce à ses nombreuses applications pour les véhicules autonomes et la robotique.

TABLE 1 – Matrice de synthèse de l'apprentissage par réseaux de neurones profonds sur des nuages de points 3D.

	PointPillars	FKAConv	PointNet++	DGCNN
Publication	CVPR 2019	ACCV 2020	CVPR 2017	AAAI 2018
Auteurs	Lang et al.	Boulch et al.	Qi et al.	Wang et al.
Type de technique	Structurée	Non structurée	Non structurée	Non structurée
Classification		✓	✓	✓
Segmentation		✓	✓	✓
Détection d'objets 3D	✓			
Type d'échantillonnage	–	EPSSQ	FPS	–
Type de regroupement	Voxélisation	KNN + Normalisation	KNN/Balle	KNN
Invariance à la permutation	✓	✓	✓	✓
Invariance à la translation	✓	✓	✓	✓
Invariance à la rotation		✓	✓	
Optimise le temps d'inférence	✓			

Références

Saifullahi Aminu Bello, Shangshu Yu, Cheng Wang, Jibril Muhmmad Adam, and Jonathan Li. Deep learning on 3d point clouds. *Remote Sensing*, 12(11) :1729, 2020.

Alexandre Boulch, Gilles Puy, and Renaud Marlet. FKAConv : Feature-Kernel Alignment for Point Cloud Convolution. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12622 LNCS :381–399, apr 2020. ISSN 16113349. doi : 10.48550/arxiv.2004.04462. URL `https://arxiv.org/abs/2004.04462v3`.

Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes : A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet : An information-rich 3d model repository. *arXiv preprint arXiv :1512.03012*, 2015.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert : Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv :1810.04805*, 2018.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words : Transformers for image recognition at scale. *arXiv preprint arXiv :2010.11929*, 2020.

Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics : The kitti dataset. *The International Journal of Robotics Research*, 32(11) :1231–1237, 2013.

Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. PointPillars : Fast Encoders for Object Detection from Point Clouds. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June : 12689–12697, dec 2018. ISSN 10636919. doi : 10.48550/arxiv.1812.05784. URL `https://arxiv.org/abs/1812.05784v2`.

Eric-Tuan Lê, Minhyuk Sung, Duygu Ceylan, Radomir Mech, Tamy Boubekeur, and Niloy J Mitra. Cpfn : Cascaded primitive fitting networks for high-resolution point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7457–7466, 2021.

Yiyi Liao, Jun Xie, and Andreas Geiger. Kitti-360 : A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *arXiv preprint arXiv :2109.13410*, 2021.

Daniel Maturana and Sebastian Scherer. Voxnet : A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928, 2015. doi : 10.1109/IROS.2015.7353481.

Lucas Nunes, Rodrigo Marcuzzi, Xieyuanli Chen, Jens Behley, and Cyrill Stachniss. Segcontrast : 3d point cloud feature representation learning through self-supervised segment discrimination. *IEEE Robotics and Automation Letters*, 2022.

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. 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, dec 2016. doi : 10.48550/arxiv.1612.00593. URL `https://arxiv.org/abs/1612.00593v2`.

Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++ : Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *Advances in Neural Information Processing Systems*, 2017-December :5100–5109, jun 2017. ISSN 10495258. doi : 10.48550/arxiv.1706.02413. URL `https://arxiv.org/abs/1706.02413v1`.

Yongming Rao, Jiwen Lu, and Jie Zhou. Spherical fractal convolutional neural networks for point cloud recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 452–460, 2019.

Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet : Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2530–2539, 2018.

Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving : Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic Graph CNN for Learning on Point Clouds. *ACM Transactions on Graphics*, 38(5) :13, jan 2018. ISSN 15577368. doi : 10.48550/arxiv.1801.07829. URL `https://arxiv.org/abs/1801.07829v2`.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets : A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

Yuxing Xie, Jiaojiao Tian, and Xiao Xiang Zhu. Linking points with labels in 3d : A review of point cloud semantic segmentation. *IEEE Geoscience and Remote Sensing Magazine*, 8(4) :38–59, 2020.

Yan Yan, Yuxing Mao, and Bo Li. Second : Sparsely embedded convolutional detection. *Sensors*, 18 (10) :3337, 2018.

Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor : Real-time 3d object detection from point clouds. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7652–7660, 2018.

Haoxuan You, Yifan Feng, Rongrong Ji, and Yue Gao. Pvnet : A joint convolutional network of point cloud and multi-view for 3d shape recognition. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 1310–1318, 2018.

Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-bert : Pre-training 3d point cloud transformers with masked point modeling. *arXiv preprint arXiv :2111.14819*, 2021.

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16259–16268, 2021.