



**INSTITUT  
POLYTECHNIQUE  
DE PARIS**

## Point Cloud Segmentation

par

**Youssef Ben Cheikh  
Andrés Eloy Rubio Carvajal  
Maxime Fantino  
Jonathan Gusching  
Esteban Rousseau  
Charles Thonier**

**Professeurs encadrants :**  
Jules Sanchez, Gianni Franchi



# Sommaire

<b>Introduction</b>	<b>4</b>
<b>1 U-Net et l'approche naïve de la segmentation sémantique</b>	<b>5</b>
1.1 Présentation . . . . .	5
1.2 Architecture . . . . .	5
1.3 Résultats . . . . .	6
1.3.1 Entraînement . . . . .	6
1.3.2 Performance sur les classes . . . . .	6
1.3.3 Résultats qualitatifs . . . . .	7
<b>2 RangeNet++ : une approche plus efficace</b>	<b>9</b>
2.1 Présentation . . . . .	9
2.2 Architecture . . . . .	9
2.3 Résultats . . . . .	10
2.4 RangeNet++ vs U-Net . . . . .	10
<b>3 Apprentissage semi-supervisé : ClassMix</b>	<b>11</b>
3.1 L'algorithme ClassMix . . . . .	11
3.2 ClassMix avec notre backbone et Semantic-Kitti . . . . .	11
<b>4 Utilisation de transformers</b>	<b>13</b>
4.1 Transformers . . . . .	13
4.2 SETR . . . . .	13
4.2.1 À propos de SETR . . . . .	13
4.2.2 Architecture du modèle . . . . .	14
4.2.3 SETR pour notre cas . . . . .	14
4.3 SegFormer . . . . .	14
4.3.1 Architecture du modèle . . . . .	14
4.3.2 Notre implémentation . . . . .	15
4.3.3 Résultats . . . . .	15
4.3.3.1 Entraînement . . . . .	15
4.3.3.2 Évaluation sur les classes . . . . .	15
4.3.3.3 Résultats qualitatifs . . . . .	16
4.3.4 Conclusion . . . . .	16
4.4 RangeViT . . . . .	17
4.4.1 Architecture du modèle . . . . .	17
4.4.2 Notre implémentation . . . . .	18
4.4.2.1 Architecture de notre modèle . . . . .	18
4.4.2.2 Entraînement . . . . .	18

4.4.2.3	Performances sur les classes et comparaison avec SegFormer	18
4.4.2.4	Résultats qualitatifs . . . . .	19
4.4.2.5	Conclusion . . . . .	20
<b>Conclusion</b>		<b>21</b>
<b>Bibliographie</b>		<b>21</b>
<b>A Note technique sur le Lidar</b>		<b>23</b>
A.1	Principe général . . . . .	23
A.2	Fonctionnement technique . . . . .	23
A.2.1	Composants . . . . .	23
A.2.2	Les différents types de LiDAR . . . . .	23
A.2.3	Principes physiques . . . . .	24
A.2.3.1	Émission . . . . .	24
A.2.3.2	Balayage . . . . .	24
A.2.3.3	Réception . . . . .	25
A.2.4	Application à l'imagerie LiDAR . . . . .	25
<b>B Note sur le dataset</b>		<b>26</b>
<b>C Note sur Range Projection</b>		<b>27</b>

# Introduction

Vous les avez vu rouler sur le campus de Palaiseau, les voitures autonomes sont en développement actif dans notre université. Pour évoluer dans son environnement, elles doivent apprendre à "voir" les objets de la route soit pour rester dessus, comme la route, soit pour ralentir comme à l'approche d'un passage piéton, soit les éviter comme un arbre. C'est pour cela que ces véhicules sont bardés de caméras et de capteurs, pour observer l'environnement. Mais de même qu'un appareil photo ne sait pas nommer ce qu'il photographie, une caméra de voiture autonome ne sait pas ce qu'elle filme et il faut donc le lui apprendre. Ce processus de division de l'image en objets s'appelle la *segmentation*. Il existe des méthodes purement liées au traitement d'image, qui vont par exemple détecter des objets par leur contour, mais aussi des méthodes d'apprentissage automatique, et c'est sur celles-ci que notre étude va se concentrer. Elles consistent à rassembler entre eux certains pixels de l'image et de leur attribuer une étiquette pour les classifier. On parle alors de segmentation *sémantique*.



FIGURE 1 – Exemple de segmentation d'image [sto]

# Chapitre 1

## U-Net et l'approche naïve de la segmentation sémantique

### 1.1 Présentation

L'une des approches possibles pour la segmentation sémantique par apprentissage automatique est d'utiliser un réseau de type U-Net [RFB15]. U-Net a été créé pour la segmentation sémantique en imagerie biomédicale, une tâche utile dans les diagnostics permettant d'identifier et d'isoler par exemple des tumeurs à partir d'images.

Les approches traditionnelles de segmentation d'images s'appuient sur des caractéristiques créées à la main et des techniques ad hoc, ce qui nécessite de nombreux réglages manuels et n'est pas facilement transposable à d'autres cas d'étude. De plus, ces techniques ont la réputation d'être complexes et mal adaptées aux images de grande taille et à haute résolution.

U-Net a ainsi été créé pour compenser ces aspects en apprenant des caractéristiques de haut niveau directement à partir des données d'image. Le réseau est conçu comme un réseau de neurones entièrement convolutif capable de traiter des images de taille arbitraire et intègre une architecture en forme de "U" qui lui permet de capturer à la fois des caractéristiques sémantiques de haut niveau et des détails d'image de bas niveau.

### 1.2 Architecture

L'architecture U-Net se compose d'un encodeur et d'un décodeur, reliés par une couche de bottleneck. L'encodeur capture le contexte de l'image d'entrée en la sous-échantillonnant (downsampling) à l'aide de couches de convolution, tandis que le décodeur utilise des couches de suréchantillonnage (upsampling) pour générer une carte de segmentation ayant les mêmes dimensions que l'image d'entrée. Le fait d'avoir une couche bottleneck reliant l'encodeur et le décodeur permet au réseau de combiner les informations sémantiques de haut niveau avec les informations spatiales de bas niveau. Le nombre de couches de convolution et de filtres peut être ajusté en fonction de la complexité de l'image d'entrée et de la résolution de sortie souhaitée. En plus, des skip-connections sont ajoutées entre les couches correspondantes de chaque côté du U afin de préserver les informations spatiales perdues au cours du processus de down-sampling. Cela permet au décodeur de produire des cartes de segmentation plus précises.

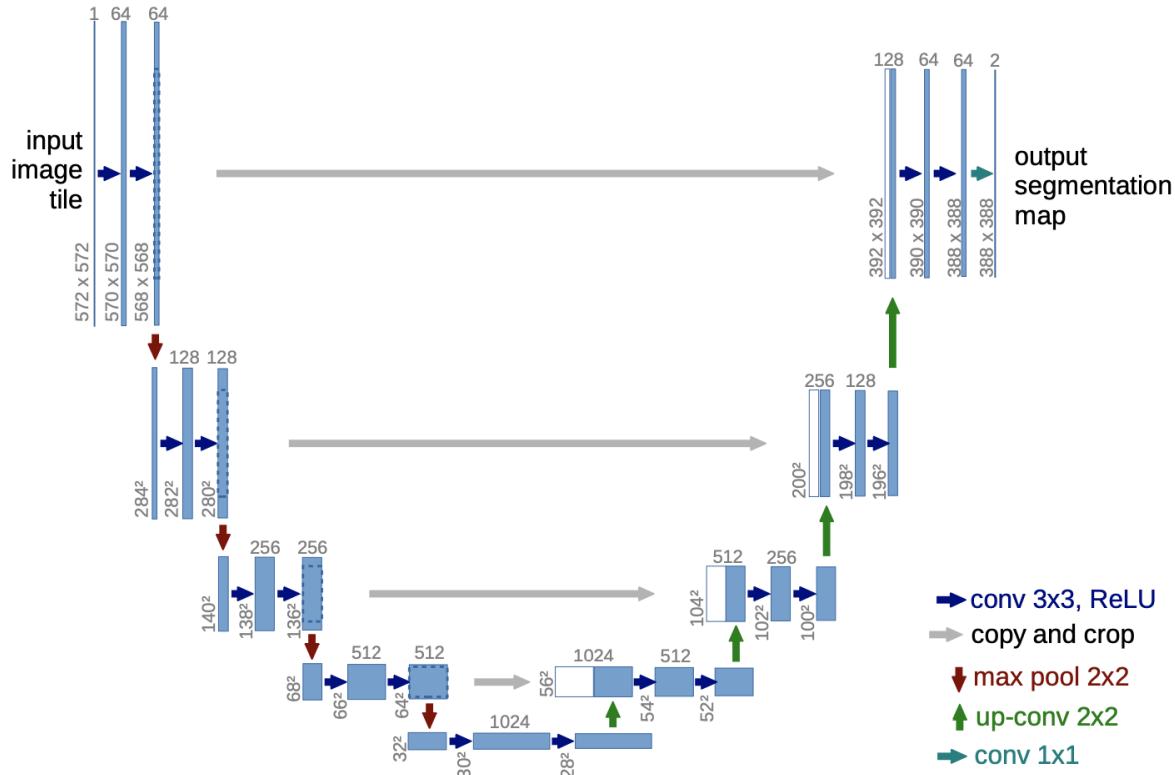


FIGURE 1.1 – Architecture du réseau U-Net [RFB15]

## 1.3 Résultats

L’implémentation du réseau U-Net appliquée au problème considéré reprend très largement le code du TP de segmentation sémantique du cours ROB313. On entraîne le modèle sur 50 epochs, en minimisant la loss CrossEntropy avec des poids pour les classes.

### 1.3.1 Entrainement

Nous avons entraîné nos modèles sur une partie du dataset SemanticKITTI, déjà projetée en range images (voir annexe C pour Range Projection). Chaque image a 5 features pour chaque pixel, x,y,z, distance et une métrique d’intensité de réflexion. On a 19 classes sur notre dataset.

Concernant l’évolution des deux métriques utilisées au cours de l’entraînement 1.2, on observe une possible difficulté de la part du réseau à généraliser correctement. A partir du vingtième epoch, les résultats atteignent une limite. Il apparaît que le modèle entre en overfitting : la loss du training continue à décroître cependant la loss de la validation reste constante voire croît, parfois. À la fin on a un mIoU=0.34.

### 1.3.2 Performance sur les classes

Les tableaux suivants présentent les meilleures IoU (intersection over union) en moyenne obtenues sur les données de test pour les différentes classes à identifier ainsi que les poids (importance de la classe) associés à ces classes. Les résultats indiquent une grande variabilité entre les classes allant de 0.00 à 0.85.

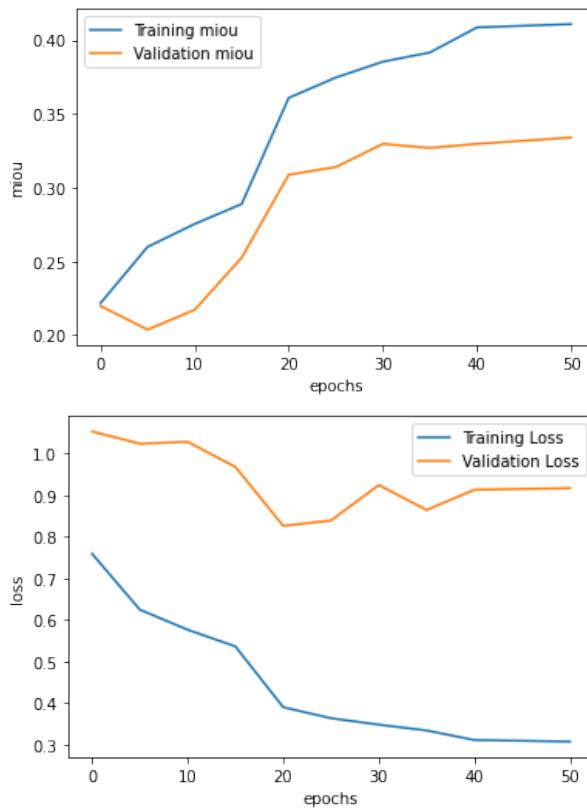


FIGURE 1.2 – Résultats d’entraînement du U-Net sur 50 epochs.

classes	car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist
IoU	0.85	0.03	0.16	0.15	0.15	0.10	0.20
poids	0.04	0.00	0.00	0.00	0.00	0.00	0.00
motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation
0.00	0.80	0.27	0.64	0.01	0.69	0.38	0.64
0.01	0.14	0.00	0.13	0.07	0.27	0.00	0.20
		trunk	terrain	pole	traffic-sign		
		0.32	0.64	0.24	0.07		
		0.01	0.08	0.00	0.00		

### 1.3.3 Résultats qualitatifs

En regardant des exemples de prédiction 1.3, on peut bien remarquer la bonne performance sur les classes : car, road, vegetation (en jaune)... Unet a pu avoir des résultats acceptables même sur les classes faiblement représentées comme fence et trunk, mais il faut s'améliorer surtout sur les classes en relation avec la conduite autonome (bicycle, person...).

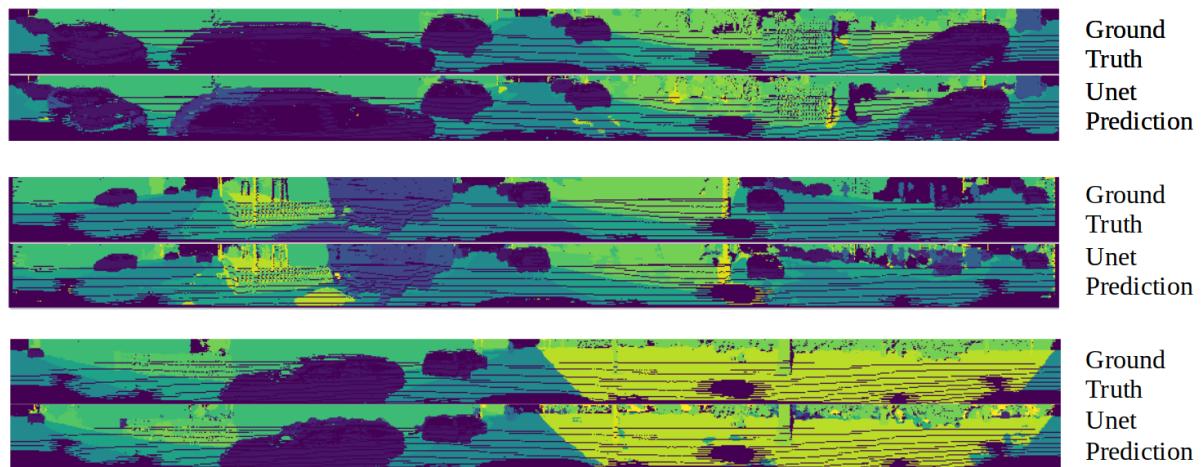


FIGURE 1.3 – Exemples de prédition avec Unet et leur group truths.

# Chapitre 2

## RangeNet++ : une approche plus efficace

### 2.1 Présentation

RangeNet++ a été proposé dans [MVBS19] pour résoudre les problèmes de segmentation sémantique en 3D des nuages de points LiDAR pour la conduite autonome. Sur une voiture, des LiDAR sont souvent utilisés pour générer des nuages de points 3D denses qui fournissent des informations sur l'environnement autour du véhicule, dont l'emplacement des autres véhicules, des piétons et des obstacles.

La segmentation sémantique de ces nuages de points consiste donc à classer chaque point dans l'une des catégories prédéfinies (voire annexe dédiée). Cependant, la segmentation sémantique 3D des nuages de points LiDAR est une tâche difficile en raison de la rareté et de l'irrégularité des données, ainsi que de la nécessité d'une estimation précise de la profondeur. Les CNN 2D traditionnels ne sont pas bien adaptés à cette tâche car ils supposent des données régulières sous forme de grille, alors que les nuages de points LiDAR sont épars et non structurés.

RangeNet++ a été créé pour résoudre ces problèmes en utilisant une architecture de Sparse Convolutional Neural Network (SCNN) spécialement conçue pour traiter ce genre de données. Le réseau intègre également des informations sur la profondeur en utilisant une représentation multicanaux du nuage de points qui code à la fois les coordonnées 3D et l'intensité du faisceau laser réfléchi. L'architecture RangeNet++ est ainsi capable d'atteindre une grande précision dans la segmentation sémantique des nuages de points.

### 2.2 Architecture

L'architecture de RangeNet++ consiste en plusieurs blocs de SCNN, de normalisation par batch et de fonctions d'activation ReLU, suivis de couches de max-pooling. La sortie de chaque bloc passe par un autre bloc avec moins de convolutions, ce qui réduit l'échantillonnage de l'entrée et augmente le nombre de channels. Cela permet au réseau de capturer des caractéristiques à différentes échelles et résolutions.

Après le dernier bloc, la sortie passe par plusieurs couches de déconvolution, qui rééchantillonnent progressivement les feature maps jusqu'à la résolution d'origine du nuage de points d'entrée. Enfin, une fonction d'activation softmax est appliquée pour générer la carte de segmentation sémantique.

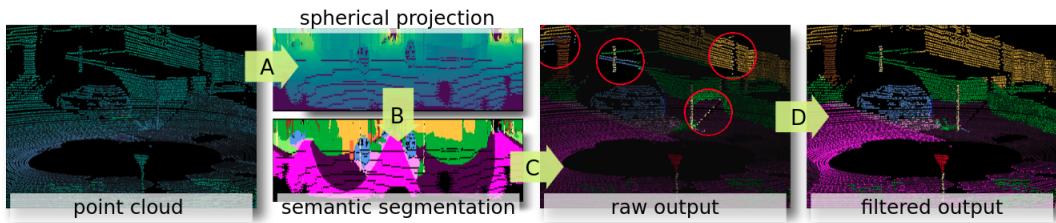


FIGURE 2.1 – La pipeline de RangeNet++. Le principe premier de RangeNet++ est l'ajout de post-processing sur des prédictions après avoir été reprojétées dans un espace en 3 dimensions plutôt que sur une image.

Il est possible de paramétriser spécifiquement le nombre de convolutions, de filtres et la taille de regroupement fonction de la taille et de la complexité des données du nuage de points d'entrée.

## 2.3 Résultats

Aucun entraînement n'a été réalisé même si nous avons essayé de le faire sur nos machines, qui se sont révélées ne pas posséder assez de RAM.

## 2.4 RangeNet++ vs U-Net

L'un des avantages de RangeNet++ par rapport à U-Net est sa capacité à traiter des données 3D et à intégrer des informations sur la profondeur, ce qui est particulièrement important pour les applications de conduite autonome. RangeNet++ utilise également une architecture différente appelée Sparse Convolutional Neural Network (SCNN). Dans la littérature, RangeNet++ est justement utilisé pour traiter les données clairsemées des nuages de points des relevés d'un capteur LiDAR.

De son côté, U-net a été conçu pour l'imagerie médicale. Il s'agit également d'une architecture simple à mettre en œuvre et à entraîner. Cependant, l'une des faiblesses potentielles du U-Net est qu'il produire de mauvais scores sur des images très grandes ou très complexes en raison des limitations de mémoire. En outre, il est a priori, par son design, moins efficace que RangeNet++ pour le traitement des données 3D.

# Chapitre 3

## Apprentissage semi-supervisé : ClassMix

Obtenir des labels pour des données issues de LIDAR est un processus long et coûteux. Cela signifie qu'avec un LIDAR, une partie des données risque dans les applications concrètes de ne pas pouvoir être labellisée à temps pour l'entraînement. Pour remédier à ce type de problèmes, on aimerait être capables d'exploiter des données sans labels : c'est le principe de l'apprentissage semi-supervisé.

### 3.1 L'algorithme ClassMix

L'algorithme ClassMix [OTPS20] repose sur un principe d'élève-enseignant où un modèle est entraîné pour avoir le rôle d'enseignant. L'entraînement se déroule alors de la manière suivante :

- Pour les images déjà labellisées, l'algorithme de segmentation est appliqué directement (en l'occurrence RangeNet++ dans notre cas, ResNet dans la publication originale)
- En ce qui concerne les images non-labellisées, c'est ici que ClassMix intervient. Deux images non labellisées sont sélectionnées au hasard et sont combinées (figure 3.1) pour en former une nouvelle. Comme montré dans la figure 3.2, on génère aussi des prédictions qui sont combinées de la même manière que les images. La combinaison se fait à partir d'un choix aléatoire de classes prédites, qui vont donner un masque pour extraire des éléments d'une image qui seront combinés à l'autre. L'intérêt du pseudo-label (c'est-à-dire, l'utilisation de pour choisir la classe la plus probable pour chaque pixel) plutôt que de garder les probabilités prédites permet d'éviter la contamination des labels, comme il est écrit dans la publication.

En pratique, l'entraînement de ClassMix s'est avéré très laborieux, étant donné la taille des données à garder en mémoire (bien plus que 8Go, ce qui ne nous a pas permis de faire des essais sur les ordinateurs personnels qui sont limités en RAM et VRAM).

### 3.2 ClassMix avec notre backbone et Semantic-Kitti

Notre objectif était de combiner l'architecture ClassMix avec RangeNet pour une utilisation sur la base de données Semantic-Kitti. Malheureusement, nous n'avons pas pu obtenir un résultat ayant abouti à partir des architectures déjà implémentées.

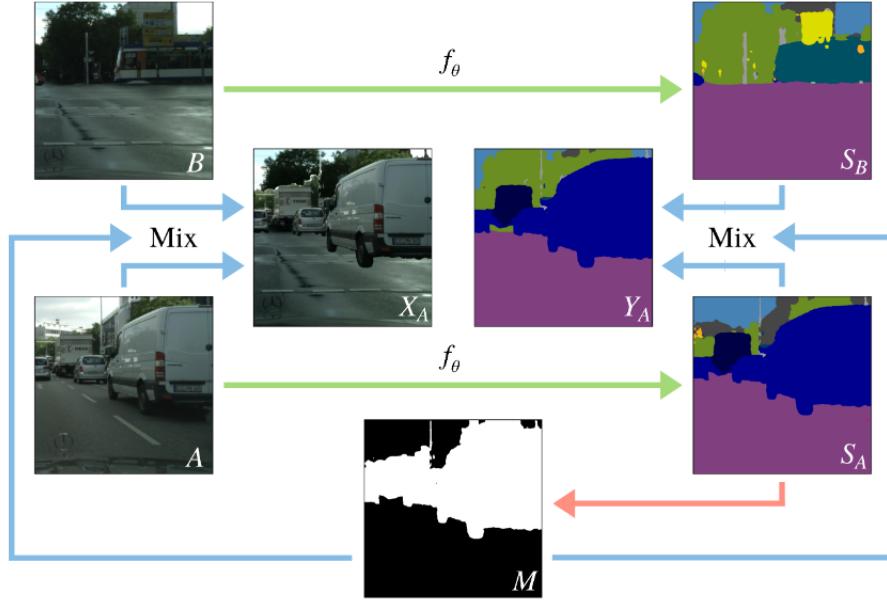


FIGURE 3.1 – Principe de data augmentation de ClassMix

**Algorithm 1** ClassMix algorithm

**Require:** Two unlabelled samples  $A$  and  $B$ , segmentation network  $f_{\theta'}$ .

- 1:  $S_A \leftarrow f_{\theta'}(A)$
- 2:  $S_B \leftarrow f_{\theta'}(B)$
- 3:  $\tilde{S}_A \leftarrow \arg \max_{c'} S_A(i, j, c') \triangleright$  Take pixel-wise argmax over classes.
- 4:  $C \leftarrow$  Set of the different classes present in  $\tilde{S}_A$
- 5:  $c \leftarrow$  Randomly selected subset of  $C$  such that  $|c| = |C|/2$
- 6: For all  $i, j$ :  $M(i, j) = \begin{cases} 1, & \text{if } \tilde{S}_A(i, j) \in c \\ 0, & \text{otherwise} \end{cases} \triangleright$  Create binary mask.
- 7:  $X_A \leftarrow M \odot A + (1 - M) \odot B \quad \triangleright$  Mix images.
- 8:  $Y_A \leftarrow M \odot S_A + (1 - M) \odot S_B \quad \triangleright$  Mix predictions.
- 9: **return**  $X_A, Y_A$

FIGURE 3.2 – L'algorithme ClassMix tel que décrit dans la publication

# Chapitre 4

## Utilisation de transformers

### 4.1 Transformers

Les transformers sont une architecture de réseau de neurones profondément innovante pour le traitement de données séquentielles telles que le langage naturel, la musique et les images. L'architecture a été introduite en 2017 par des chercheurs de Google et a été largement adoptée dans de nombreux domaines.

La clé de l'efficacité des transformers est leur mécanisme d'attention. L'attention permet à l'algorithme de se concentrer sur les parties les plus importantes de l'entrée séquentielle pour la tâche en cours, au lieu de traiter l'ensemble de la séquence en même temps. Cette approche est très différente des modèles de traitement séquentiel précédents, qui examinaient chaque élément de la séquence de manière séquentielle et dans le même ordre.

Les transformers ont été initialement utilisés pour la traduction automatique, où ils ont surpassé les modèles précédents en termes de qualité de la traduction et de vitesse d'exécution. Depuis lors, les transformers ont été appliqués avec succès à de nombreux autres problèmes de traitement du langage naturel, tels que la classification de texte, la compréhension de texte et la génération de texte.

Les transformers ont également été utilisés avec succès dans d'autres domaines, tels que la vision par ordinateur et le traitement du son, où ils ont aidé à atteindre des résultats de pointe dans des tâches telles que la reconnaissance d'objets, la segmentation d'image et la génération de musique.

### 4.2 SETR

#### 4.2.1 À propos de SETR

SETR (SEgmentation TRansformer, [XWY<sup>+</sup>21]) est un modèle d'apprentissage profond utilisé pour la segmentation sémantique d'images. Il utilise une variante de l'architecture de Transformer appelée "transformer spatial" pour prendre en compte les relations spatiales entre les différentes parties de l'image. SETR utilise également une méthode d'attention dite "axiale" pour modéliser les relations à longue distance entre les parties de l'image, et un module de compression de l'information pour réduire la quantité d'informations à traiter. En combinant ces techniques, SETR est capable de capturer des informations contextuelles importantes pour une segmentation précise et efficace des images.

### 4.2.2 Architecture du modèle

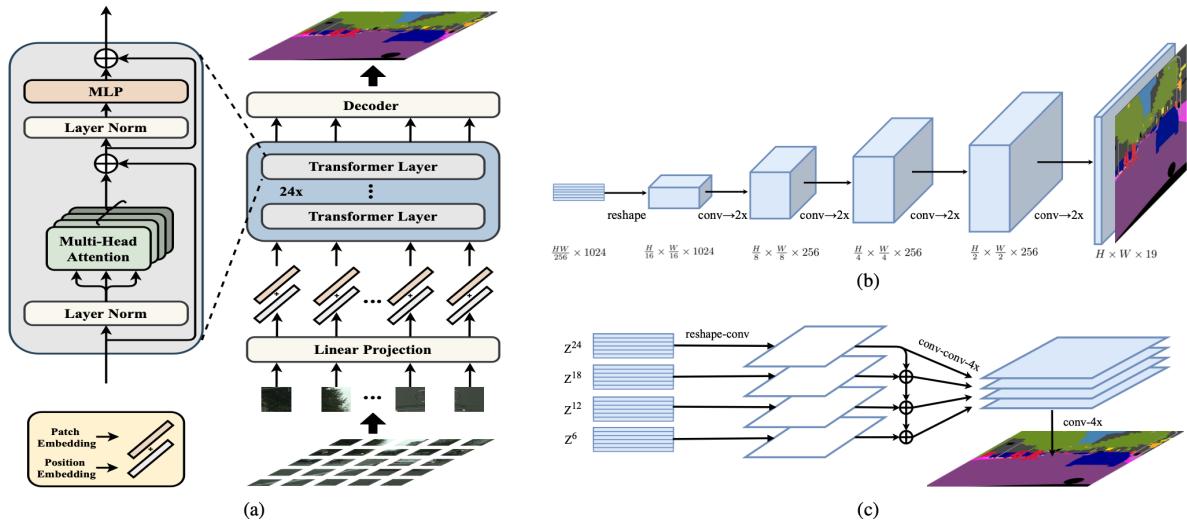


FIGURE 4.1 – Architecture de SETR. (a) L'image est éclatée en des vignettes de taille fixe qui sont traitées par un transformeur. Ensuite il y a deux décodeurs possibles : (b) upsampling progressif ou (c) agrégation de features de plusieurs niveaux. [ZLZ<sup>+</sup>21]

### 4.2.3 SETR pour notre cas

Nous avons essayé d'adapter un modèle SETR pour faire la segmentation sémantique sur la dataset RangeProjection, en spécifiant les nombres de channels d'entrée et le nombre de classes. Après nous avons entraîné le modèle durant 25 epochs. Malheureusement, le loss ne se réduisait pas et les mIoUs restaient pratiquement à 0 pour l'entraînement et pour l'évaluation.

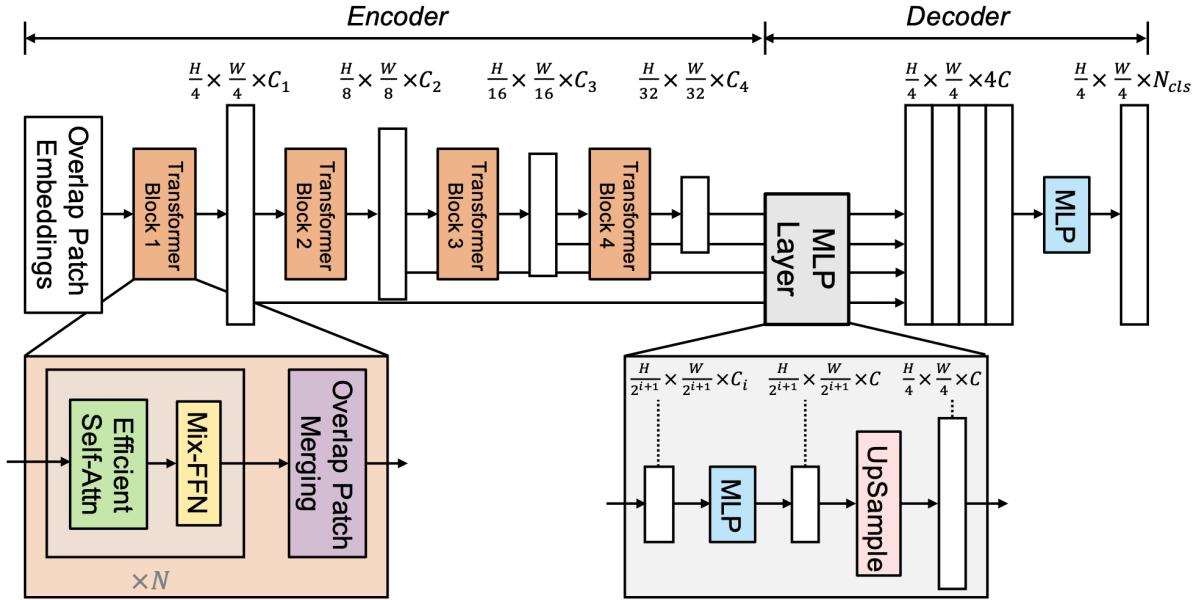
Nous n'avions pas suffisamment de temps pour investiguer la cause de la faible performance, mais cela peut être dû au fait que SETR a été spécifiquement conçu pour la segmentation sémantique d'images et que Les nuages de points de LIDAR, en revanche, sont des données tridimensionnelles et éparses qui nécessitent une représentation et une compréhension différente de l'espace.

## 4.3 SegFormer

Nous avons étudié le framework de segmentation sémantique SegFormer [XWY<sup>+</sup>21]. Il associe un encodeur transformeur et un perceptron multi-couches (MLP) à la place d'un décodeur. L'encodeur produit des features à différentes échelles et le MLP combine les informations des différentes couches ce qui permet de combiner les informations locales et globales pour produire le masque de segmentation sémantique.

### 4.3.1 Architecture du modèle

Voir figure 4.2.

FIGURE 4.2 – Architecture de SegFormer. [XWY<sup>+</sup>21]

### 4.3.2 Notre implémentation

On charge un modèle de segformer pré-entraîné sur ade20k. On change la première Convolution dans le patch embedder pour prendre en compte les 5 features, et le head classifier pour travailler sur les 20 classes de notre data set.

Il faut noter que segformer n'utilise pas d'embedding positionnel, au contraire de SETR. Segformer divise l'image en des patchs de 4x4, et fait la classification pour chaque patch, donc si la taille de l'image de l'entrée est  $W * H$ , la taille du semantic mask sera  $\frac{W}{4} * \frac{H}{4}$ .

### 4.3.3 Résultats

#### 4.3.3.1 Entraînement

On entraîne aussi sur 25 epochs. Avec des poids pour les classes dans la fonction de loss CrossEntropy (pour remédier au déséquilibre de ciel et de route par rapport aux autres classes). Nos résultats sont reportés sur les courbes de la Figure 4.3. On a obtenu un mIoU de 0.30.

On remarque bien que la loss diminue et le mIoU augmente pour le training et le test. La loss a l'air de continuer à diminuer si on continue le training (chose qui n'a pas été faite à cause des limitations en ressources et en temps).

#### 4.3.3.2 Évaluation sur les classes

On donne l'IoU sur chaque classe :

classes	car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist
IoU	0.80	0.01	0.08	0.13	0.14	0.07	0.13
poids	0.04	0.00	0.00	0.00	0.00	0.00	0.00

motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation
0.00	0.76	0.21	0.57	0.00	0.65	0.37	0.59
0.01	0.14	0.00	0.13	0.07	0.27	0.00	0.20

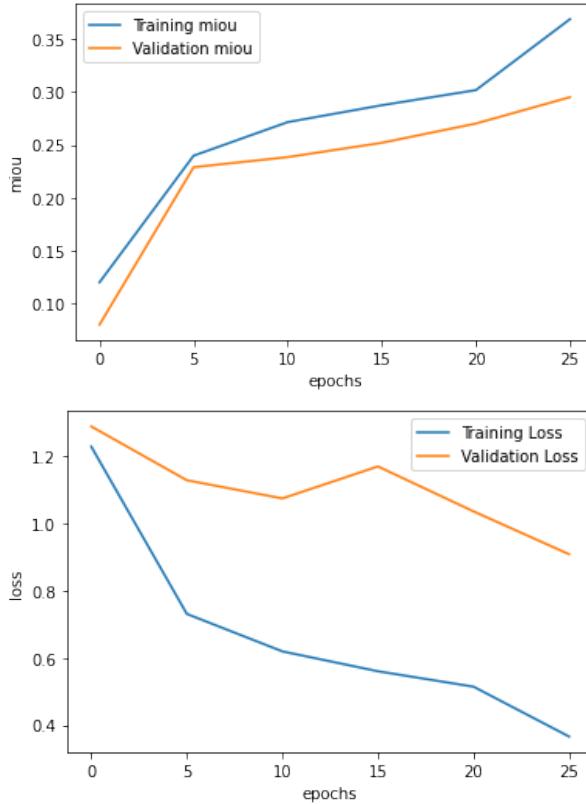


FIGURE 4.3 – Résultats d’entraînement de segformer sur 25 epochs.

trunk	terrain	pole	traffic-sign
0.26	0.59	0.20	0.06
0.01	0.08	0.00	0.00

On constate que le modèle a de bonnes performances sur certaines classes telles que car et road mais moyennes voire faibles sur les autres classes.

On remarque une tendance à avoir de bonnes performances pour les classes les plus présentes sur le dataset (building, road, vegetation, sidewalk), mais cela n'a pas empêché des classes moins représentées d'éviter de mauvaises performances (fence, terrain, car). On peut dire que c'est grâce au weightning dans la fonction du loss.

#### 4.3.3.3 Résultats qualitatifs

Regardons quelques exemples de prédictions dans la figure 4.3.

On peut bien remarquer la perte de résolution à cause de la dimension de sortie réduite de segformer. Cette perte de résolution peut expliquer la faible performance sur certaines classes comme person et bicyclist. À cause de leur morphologie, ils seront toujours minoritaire dans les patchs 4\*4.

#### 4.3.4 Conclusion

On peut voir la bonne segmentation des voitures et des routes. Mais SegFormer peut ne pas être capable de détecter les personnes et les vélos ce qui reste problématique surtout pour des tâches en relation avec la conduite autonome.

Une approche d'amélioration possible serait d'essayer de rendre l'output de la même taille que l'input.

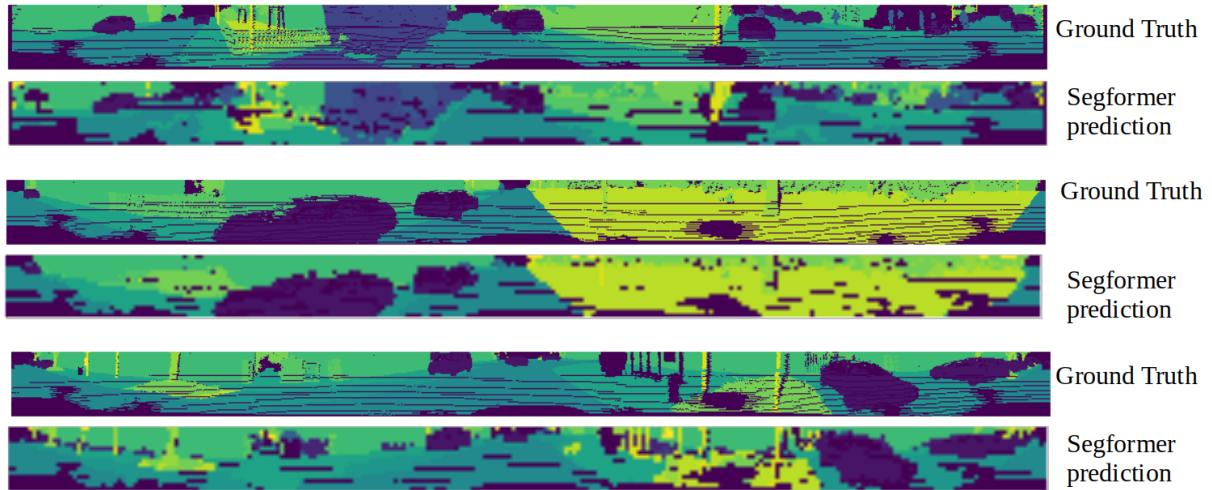


FIGURE 4.4 – Exemples de prédiction avec segformer et leur ground truth.

## 4.4 RangeViT

Ce papier [AGB<sup>+</sup>23] explore la possibilité d'exploiter les transformers dans la tâche de segmentation sémantique des 3D LiDAR point clouds.

### 4.4.1 Architecture du modèle

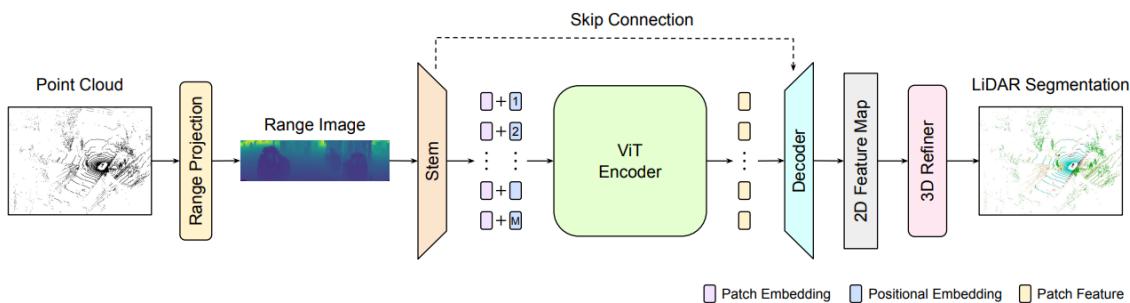


FIGURE 4.5 – Architecture du RangeViT. [AGB<sup>+</sup>23]

- **Range Projection** : L'entrée du modèle est une représentation en 2D du nuage de points. La fameuse Range Projection est utilisé pour transformer les données en 3D en une projection en 3D.
- **Convolutional stem** : Convolutional Stem (ou en français, tige convulsive) est un terme utilisé pour décrire la première couche d'un réseau de neurones convolutifs (CNN). Le convolutional stem vient remplacer la division de l'image en patchs, qui seront "linearly embedded". Cette méthode n'est pas trouvée optimale pour la tâche et le dataset en question. Cependant il est démontré que la convolution non linéaire est plus stable et améliore les performances du modèle.

- **ViT encoder** : La sortie du convolutional stem peut être donnée directement à l'encodeur ViT. On ajoute alors un token pour la classe et on additionne avec les "positional embeddings". L'encoder utilisé dans le papier est ViT-S/16, modèle pré-entraîné sur ImageNet21k pour la classification et fine-tuned après sur Cityscapes pour la segmentation sémantique des images.
- **Decoder** qui fournit des feature maps de la même résolution que l'image en profondeur d'origine.
- **3D Refiner** Convertit pixel-par-pixel les caractéristiques de l'image de profondeur en une prédiction point-par-point dans l'espace 3D.

#### 4.4.2 Notre implémentation

On travaille sur une range projection prête, donc nous n'avons pas eu besoin de la projeter.

##### 4.4.2.1 Architecture de notre modèle

- **Convolutional Stem** : on reproduit la même architecture du stem utilisé dans le papier en prenant les valeurs des paramètres optimales trouvées dans les tests (ie. nombres de canaux).
- **ViT encoder** : On adapte le code source de ViT B/16 de PyTorch. Donc on a donc un transformer qui n'est pas pré-entraîné.
- **Decoder** : On reproduit la même architecture de Decoder adopté dans le papier.
- **Segmentation Head** : Au lieu du 3D Refiner, on met un segmentation head pour générer les segmentations en 2D.

##### 4.4.2.2 Entraînement

On entraîne sur 50 epochs, avec des poids des classes aussi comme pour SegFormer, à la fin on obtient un mIoU = 0.15 4.6.

On remarque qu'au début la loss diminue pour l'entraînement et la validation, mais vers l'epoch 10, la même chose continue pour le training, mais pour la validation la loss augmente et le mIoU diminue. Il apparaît que le modèle entre en sur-apprentissage après l'epoch 10.

##### 4.4.2.3 Performances sur les classes et comparaison avec SegFormer

Performance de RangeViT et de Segformer par classe :

classes	car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	
Segformer	0.80	0.01	0.08	0.13	0.14	0.07	0.13	
RangeViT	0.59	0.00	0.01	0.03	0.04	0.03	0.05	
motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation	
0.00	0.76	0.21	0.57	0.00	0.65	0.37	0.59	
0.00	0.62	0.00	0.24	0.01	0.45	0.08	0.23	
trunk		terrain	pole	traffic-sign				
		0.26	0.59	0.20	0.06			
		0.05	0.15	0.19	0.00			

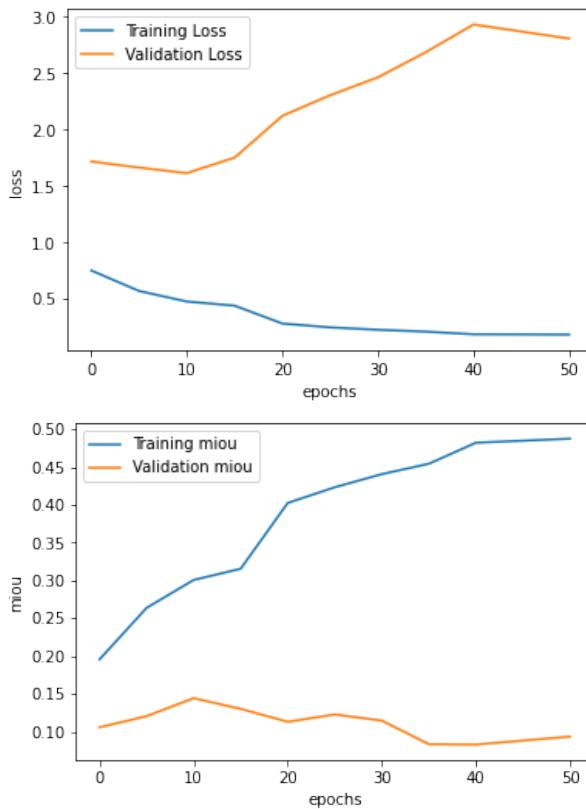


FIGURE 4.6 – Résultats d’entraînement de RangeViT sur 50 epochs.

On remarque que SegFormer surpassé les performances de RangeViT sur à peu près toutes les classes. RangeViT a des bonnes performances sur car, road et buildings. Il y a des classes qui posent des problèmes pour les deux modèles : comme person et bicyclist.

#### 4.4.2.4 Résultats qualitatifs

4.7 RangeViT conserve la résolution. On voit une bonne performance sur cars et roads.

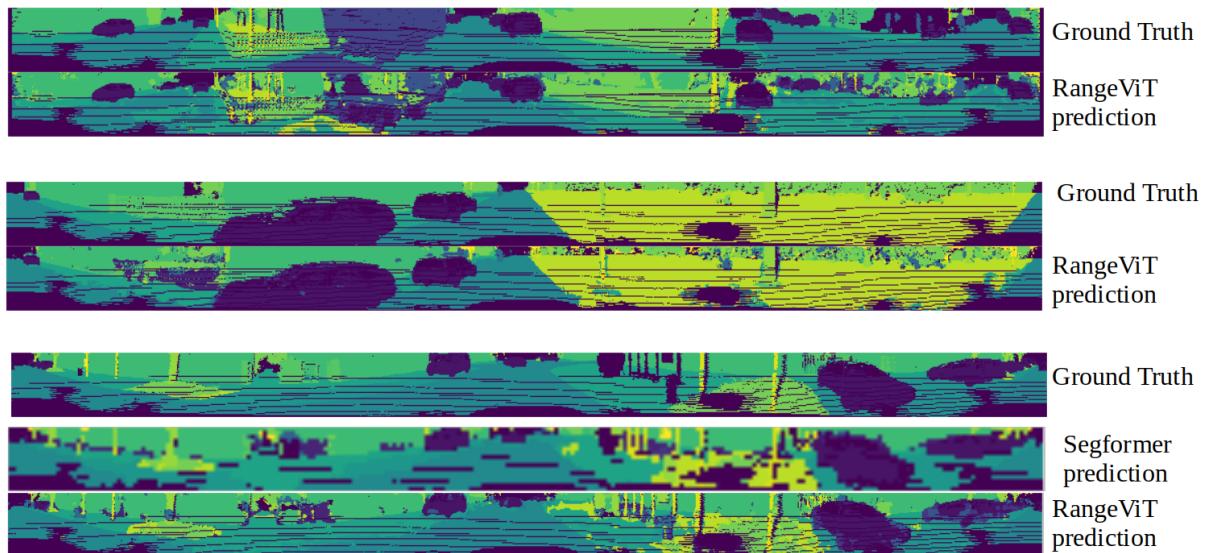


FIGURE 4.7 – Exemples de prédiction avec RangeViT et leur group truths.

#### 4.4.2.5 Conclusion

Pour surmonter la difficulté de l'entraînement du ViT, on peut entraîner sur un dataset plus large ou bien utiliser un ViT pré-entraîné (comme c'est fait dans le papier Range ViT).

Unet a eu des résultats plus bon que nos modèles de transformers, ceci peut se justifier par l'architecture simple de Unet et son petit nombre de paramètres entraînables, ce qui facilite l'entraînement sur des petits datasets. (On entraîne seulement sur une sub dataset extraite de semantickitti)

# Conclusion

Le réseau présenté en première partie est le U-Net, une architecture assez simple. Un modèle de cette forme a été entraîné et les résultats obtenus étaient bons pour certaines classes et médiocres pour d'autres.

Une étude de l'architecture globale explicité dans le papier RangeNet++ (état de l'art pour la segmentation avec données LiDAR) a également été réalisée même si nous n'avons pas réalisé nos propres entraînements pour cette dernière.

Comme rappelé précédemment, la labellisation de données en segmentation sémantique nécessite énormément de temps puisque chaque image est constituée de plusieurs milliers de points à labelliser un par un. Afin d'éviter d'avoir à faire ce travail à la main, nous nous sommes penchés sur l'apprentissage semi-supervisé et plus précisément sur l'algorithme ClassMix. L'objectif initial était de combiner ClassMix et RangeNet++ afin d'accroître la qualité des résultats. Cependant nous n'avons pas pu le mettre en application en raison de notre absence de résultats pour RangeNet++.

Pour finir nous avons exploré une voie différente en utilisant des transformers. Nous avons essayé d'implémenter trois architectures différentes. Tout d'abord le modèle SETR qui éclate l'image en vignettes de taille fixe sur le dataset RangeProjection. Toutefois il semblerait que ce modèle ne soit pas adapté pour des images 3D Lidar. L'approche SegFormer a eu quant à elle plus de succès avec une loss qui continuait à décroître au fur et à mesure des epochs, avec de bonnes performances notamment sur les classes car et road tout en évitant de mauvaises performances sur les classes les moins présentes du dataset, mais ça reste en dessous des performances d'Unet. Enfin, nous avons étudié le papier RangeViT qui rejoignait notre projet puisqu'il propose d'utiliser des transformers pour la segmentation sémantique d'image 3D LiDAR. Nous avons pu coder notre propre implémentation de ce papier qui nous a conduit à des résultats moins bons que ceux de SegFormer avec un état de sur-apprentissage au bout d'une dizaine d'epochs.

Ce projet nous a donc permis d'explorer plusieurs approches de Deep Learning pour diviser l'espace en objets et de comparer leurs performances sur un sujet de recherche très actif et proche de nous, puisque les véhicules autonomes roulent déjà sur le campus.

# Bibliographie

- [2] Qu'est-ce que la technologie lidar ? <https://www.generationrobots.com/blog/fr/qu-est-ce-que-la-technologie-lidar/>.
- [AGB<sup>+</sup>23] Angelika Ando, Spyros Gidaris, Andrei Bursuc, Gilles Puy, Alexandre Boulch, and Renaud Marlet. Rangevit : Towards vision transformers for 3d semantic segmentation in autonomous driving, 2023.
- [BGM<sup>+</sup>19] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI : A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019.
- [CTHB16] Patrick Chazette, Julien Totems, Laurent Hespel, and Jean-Stéphane Bailly. *Principle and Physics of the LiDAR Measurement*, pages 201–247. 12 2016.
- [MVBS19] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. Rangenet ++ : Fast and accurate lidar semantic segmentation. pages 4213–4220, 11 2019.
- [OTPS20] Viktor Olsson, Wilhelm Tranheden, Juliano Pinto, and Lennart Svensson. Classmix : Segmentation-based data augmentation for semi-supervised learning. *CoRR*, abs/2007.07936, 2020.
- [RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net : Convolutional networks for biomedical image segmentation, 2015.
- [sto] Semantic segmentation autonomous wheelchair. <https://github.com/VenkatNarayanan11/Semantic-Segmentation-Autonomous-Wheelchair>.
- [str] Pushing the limits of learning-based traversability analysis for autonomous driving on cpu. [https://www.researchgate.net/publication/361161461\\_Pushing\\_the\\_Limits\\_of\\_Learning-based\\_Traversability\\_Analysis\\_for\\_Autonomous\\_Driving\\_on\\_CPU](https://www.researchgate.net/publication/361161461_Pushing_the_Limits_of_Learning-based_Traversability_Analysis_for_Autonomous_Driving_on_CPU).
- [XWY<sup>+</sup>21] Enze Xie, Wenhui Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer : Simple and efficient design for semantic segmentation with transformers. In *Neural Information Processing Systems (NeurIPS)*, 2021.
- [ZLZ<sup>+</sup>21] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip H. S. Torr, and Li Zhang. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers, 2021.

# Annexe A

## Note technique sur le Lidar

### A.1 Principe général

Un LiDAR est un capteur permettant de récolter des données sur l'environnement d'un système à l'instar d'un radar ou d'un sonar. LiDAR est l'acronyme de *Light Detection And Ranging*, ainsi un tel capteur utilise un faisceau laser pour déterminer la distance entre un obstacle visé et lui-même.

Un LiDAR est dit capteur de temps de vol, c'est-à-dire qu'il mesure le temps de parcours du faisceau lumineux émis pour déterminer la distance de l'obstacle.

### A.2 Fonctionnement technique

#### A.2.1 Composants

Comme évoqué précédemment, on peut grossièrement décomposer un capteur LiDAR en deux éléments principaux :

- Un laser qui servira de source lumineuse.
- Un capteur d'image qui utilisera les rayons réfractés pour cartographier l'environnement.

En pratique la chaîne fonctionnelle est plus complexe et comporte bien plus d'éléments.

#### A.2.2 Les différents types de LiDAR

Les capteurs LiDAR peuvent être divisés en quatre sous-catégories :

- Le type le plus commun, qui est utilisé dans notre projet, n'utilise pas de changement de longueur d'onde. Ils sont principalement utilisés pour effectuer des mesures sur des aérosols ou des nuages situés dans l'atmosphère, mais peuvent aussi servir à détecter des surfaces.
- Certains LiDAR sont utiles pour déterminer les compositions physico-chimiques de certains milieux. Ils utilisent pour cela le fait que la longueur d'onde de la lumière réfléchie par le milieu est légèrement modifiée par la composition de ce dernier.
- Une autre catégorie est celle des differential absorption systems (DIAL). Dans ces systèmes le laser émet de la lumière à deux longueurs d'ondes très proches, dont

une est fortement absorbée par la cible. On utilise ensuite le ratio d'énergie entre les deux ondes réfléchies pour déterminer la concentration moléculaire du composé mesuré dans le milieu cible.

- La dernière grande catégorie de LiDAR est celle des systèmes hétérodynes, qui constituent la majorité des LiDAR Doppler qui servent notamment à mesurer la vitesse du vent. Ils utilisent pour cela le décalage des longueurs d'ondes réfléchies par les particules. Ce décalage est induit par la vitesse de ces particules et le mesurer permet donc de déterminer cette vitesse.[CTHB16]

### A.2.3 Principes physiques

Un capteur LiDAR utilise les lois de l'optique géométrique pour déterminer les positions des obstacles par rapport au capteur lui-même. À la base de toute mesure LiDAR se trouve une impulsion laser [CTHB16]. À la fin de la chaîne optique se trouve un capteur photosensible qui convertit le signal optique reçu en signal électrique qui est finalement numérisé.

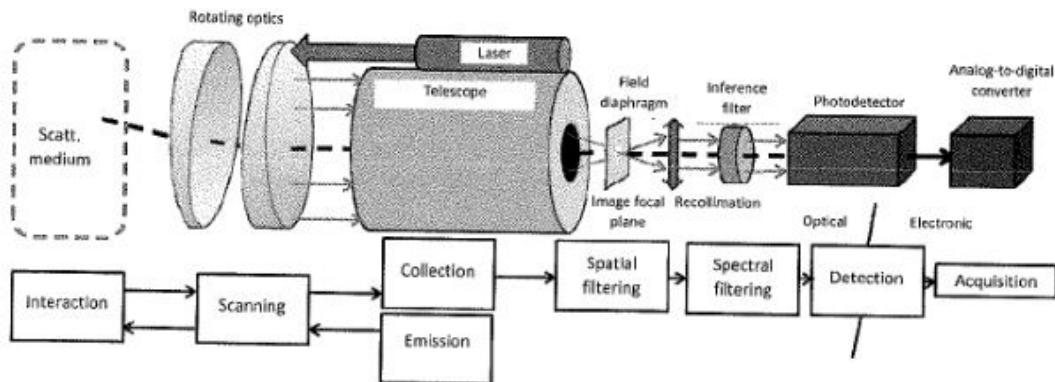


FIGURE A.1 – Architecture conventionnelle d'un capteur LiDAR ainsi que la chaîne fonctionnelle associée.[CTHB16]

#### A.2.3.1 Émission

Le faisceau laser est généré dans une cavité puis dirigé vers les cibles à détecter. La longueur d'onde du laser peut appartenir à l'infrarouge proche, au visible ou à l'ultraviolets proche. Par rotation du support, le laser ainsi projeté dans toutes les directions de l'espace se doit d'être sûr pour les passants. En vue de réduire sa dangerosité, le faisceau est donc légèrement diffracté en sortie de cavité. Les lois de l'optique nous apprennent qu'en augmentant la largeur du faisceau son énergie surfacique diminue.

#### A.2.3.2 Balayage

Pour balayer un espace à 360°, le support du lidar doit effectuer une rotation sur lui-même. Cela permet de scanner un cylindre de petite hauteur tout autour du support. Mais pour augmenter la hauteur de ce cylindre, un stratagème doit être adopté. Il consiste à faire tourner une lentille de forme biseautée. Le chemin optique du laser en sortie de sa

cavité s'en trouve ainsi modifié et se voit décrire un cône. C'est la section de ce cône au niveau des cibles qui devient la hauteur du cylindre de notre mesure de l'environnement.

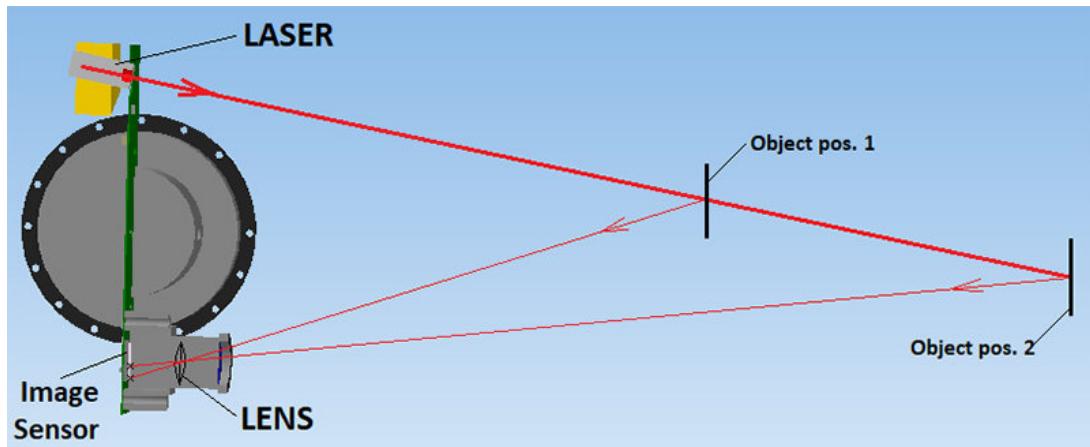


FIGURE A.2 – Schéma de principe d'un capteur LiDAR [2]

#### A.2.3.3 Réception

Après un rebond sur la ou les cibles, le rayon laser se voit dispersé. Ce faisceau réfléchi étant de grande taille, il est capté par un instrument optique également de grande dimensions, comme un télescope en montage coaxial au laser. Le signal reçu étant parasité, il convient de retirer le bruit par un filtrage spatial, puis un filtrage spectral. Le filtrage spatial a pour but de restreindre le champ de vision à l'espace parcouru par le faisceau laser, le filtrage spectral servant quant à lui à restreindre la bande spectrale autour de la longueur d'onde émise.

Le fait de connaître la position et l'orientation du laser et du capteur nous permet de déterminer les coordonnées des surfaces réfléchissantes et donc des obstacles.

#### A.2.4 Application à l'imagerie LiDAR

L'objectif d'un capteur LiDAR est de regrouper les données collectées en une image, permettant de rendre compte de l'environnement dans lequel est placé ce capteur.

# Annexe B

## Note sur le dataset

Pour nos entraînements, nous avons utilisé le jeu de données SemanticKITTI [BGM<sup>+</sup>19], qui compile les relevés Lidar d'un véhicule autonome. Le jeu de données se divise en un jeu d'entraînement et de test, chacun composé de 11 séquences richement annotées. Le dataset contient 28 classes dont la taille est représentée ci-dessous.

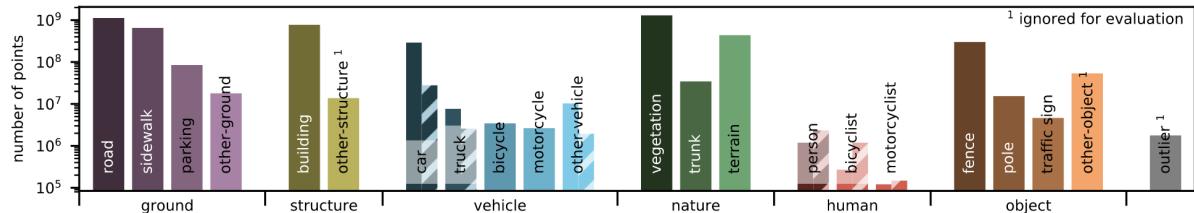


FIGURE B.1 – Les classes du jeu de données SemanticKitti [BGM<sup>+</sup>19]

Pour ce qui est de la visualisation des résultats, nous avons utilisé le logiciel Cloud Compare. Une étape de conversion était nécessaire pour convertir les données en un format compatible avec CloudCompare. Pour cela, un snippet de code nous a été partagé par notre professeur encadrant, Jules Sanchez.

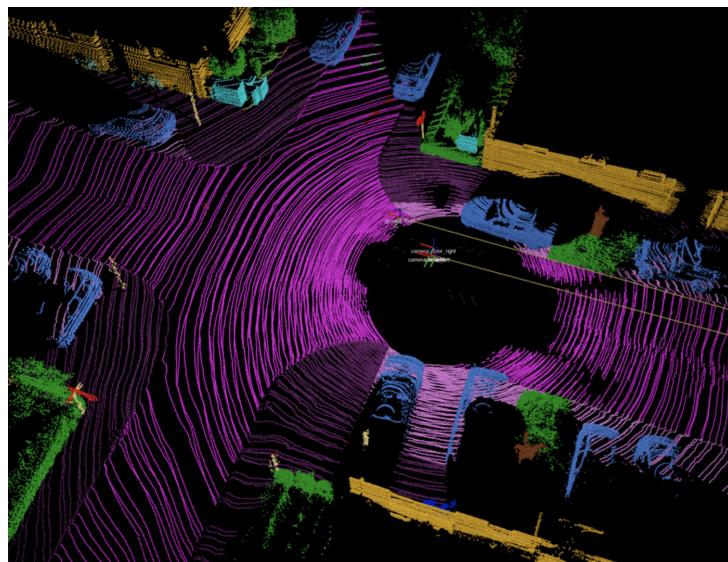


FIGURE B.2 – Visualisation des données de SemanticKitti [str]

## Annexe C

# Note sur Range Projection

La projection de portée est une méthode bien connue pour projeter chaque point dans un nuage de points sur une image de portée de taille  $H \times W$ . Les coordonnées 2D projetées  $h$  et  $w$  sont calculées en utilisant la fonction arctangente inverse et l'arcsinus inversé pour prendre en compte les dimensions de la portée et du champ de vision vertical de la caméra LIDAR.

Plus précisément, pour chaque point  $p \in P$  avec des coordonnées  $(x, y, z)$ , les coordonnées 2D projetées sont calculées à l'aide de la formule suivante :

$$[h \ w] \left[ \frac{1}{2} \frac{1 - \arctan(\frac{y}{x})}{\pi} \frac{1 - (\arcsin(\frac{z}{r}) + |f_{down}|)}{f_v} \right] [W \ H]$$

où  $r$  est la distance du point  $p$  au capteur LIDAR,  $f_{down}$  et  $f_{up}$  sont les angles de vue verticaux vers le bas et vers le haut de la caméra LIDAR, respectivement, et  $f_v$  est le champ de vision vertical total de la caméra. Cette projection de portée est effectuée pour chaque point et l'image résultante est une représentation 2D de la scène LIDAR.

Ainsi on associe cinq caractéristiques (distance, coordonnées  $x$ ,  $y$ ,  $z$  et intensité) à chaque point projeté. Si plusieurs points sont projetés sur le même pixel, seule la caractéristique du point avec la portée la plus petite est conservée. Les pixels qui ne contiennent aucun point projeté ont des caractéristiques nulles.