RangeViT: Towards Vision Transformers for 3D Semantic Segmentation in Autonomous Driving

Angelika Ando, Spyros Gidaris, Andrei Bursuc, Gilles Puy, Alexandre Boulch, Renaud Marlet

RangeViT: Towards Vision Transformers for 3D Semantic Segmentation in Autonomous Driving

Angelika Ando[1,2,*], Spyros Gidaris[1], Andrei Bursuc[1], Gilles Puy[1], Alexandre Boulch[1], Renaud Marlet[1,3]

[1]Valeo.ai, Paris, France [2]Centre for Robotics, Mines Paris, Université PSL, Paris, France
[3]LIGM, Ecole des Ponts, Univ. Gustave Eiffel, CNRS, Marne-la-Vallée, France

Abstract

Casting semantic segmentation of outdoor LiDAR point clouds as a 2D problem, e.g., via range projection, is an effective and popular approach. These projection-based methods usually benefit from fast computations and, when combined with techniques which use other point cloud representations, achieve state-of-the-art results. Today, projection-based methods leverage 2D CNNs but recent advances in computer vision show that vision transformers (ViTs) have achieved state-of-the-art results in many image-based benchmarks. In this work, we question if projection-based methods for 3D semantic segmentation can benefit from these latest improvements on ViTs. We answer positively but only after combining them with three key ingredients: (a) ViTs are notoriously hard to train and require a lot of training data to learn powerful representations. By preserving the same backbone architecture as for RGB images, we can exploit the knowledge from long training on large image collections that are much cheaper to acquire and annotate than point clouds. We reach our best results with pre-trained ViTs on large image datasets. (b) We compensate ViTs' lack of inductive bias by substituting a tailored convolutional stem for the classical linear embedding layer. (c) We refine pixel-wise predictions with a convolutional decoder and a skip connection from the convolutional stem to combine low-level but fine-grained features of the the convolutional stem with the high-level but coarse predictions of the ViT encoder. With these ingredients, we show that our method, called RangeViT, outperforms existing projection-based methods on nuScenes and SemanticKITTI. The code is available at https://github.com/valeoai/rangevit.

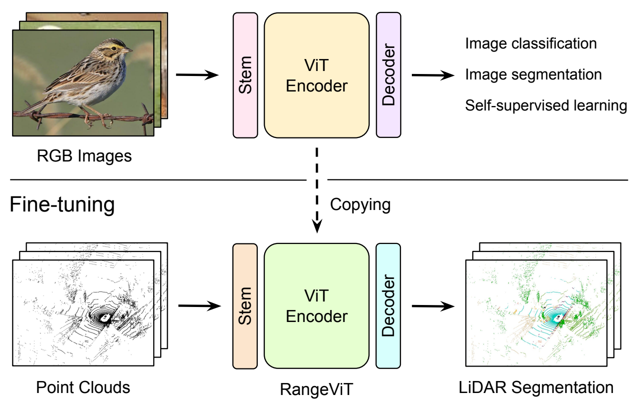

Figure 1. **Exploiting vision transformer (ViT) architectures and weights for LiDAR point cloud semantic segmentation.** We leverage the flexibility of transformer-based architectures to re-purpose them with minimal changes for processing sparse point clouds in autonomous driving tasks. The common ViT backbone across modalities allows to effectively transfer weights pre-trained on large image repositories towards improving point cloud segmentation performance with fine-tuning.

1. Introduction

Semantic segmentation of LiDAR point clouds permits vehicles to perceive their surrounding 3D environment independently of the lighting condition, providing useful information to build safe and reliable vehicles. A common approach to segment large scale LiDAR point clouds is to project the points on a 2D surface and then to use regular CNNs, originally designed for images, to process the projected point clouds [1, 11, 26, 36, 60, 66]. Recently, Vision Transformers (ViTs) were introduced as an alternative to convolutional neural networks for processing images [14]: images are divided into patches which are linearly embedded into a high-dimensional space to create a sequence of visual tokens; these tokens are then consumed by a pure transformer architecture [51] to output deep visual representations of each token. Despite the absence of almost any domain-specific inductive bias apart from the image tokenization process, ViTs have a strong representation learning capacity [14] and achieve excellent results on various

[*]This project was done during an internship at Valeo.ai.

image perception tasks, such as image classification [14], object detection [8] or semantic segmentation [45].

Inspired by this success of ViTs for image understanding, we propose to implement projection-based LiDAR semantic segmentation with a pure vision transformer architecture at its core. Our goals are threefold in doing so: (1) Exploit the strong representation learning capacity of vision transformer for LiDAR semantic segmentation; (2) Work towards unifying network architectures used for processing LiDAR point clouds or images so that any advance in one domain benefits to both; (3) Show that one can leverage ViTs pre-trained on large-size natural image datasets for LiDAR point cloud segmentation. The last goal is crucial because the downside of having few inductive biases in ViTs is that they underperform when trained from scratch on small or medium-size datasets and that, for now, the only well-performing pre-trained ViTs [9, 14, 45] publicly available are trained on large collections of images that can be acquired, annotated and stored easier than point clouds.

In this context, our main contribution is a ViT-based LiDAR segmentation approach that compensates ViTs' lack of inductive biases on our data and that achieves state-of-the-art results among projection-based methods. To the best of our knowledge, although works using ViT architectures on dense indoor point clouds already exists [63, 67], this is the first solution using ViTs for the LiDAR point clouds of autonomous driving datasets, which are significantly sparser and noisier than the dense depth-map-based points clouds found in indoor datasets. Our solution, RangeViT, starts with a classical range projection to obtain a 2D representation of the point cloud [11, 26, 36, 60]. Then, we extract patch-based visual tokens from this 2D map and feed them to a plain ViT encoder [14] to get deep patch representations. These representations are decoded using a lightweight network to obtain pixel-wise label predictions, which are projected back to the 3D point cloud.

Our finding is that this ViT architecture needs three key ingredients to reach its peak performance. First, we leverage ViT models pre-trained on large natural image datasets for LiDAR segmentation and demonstrate that our method benefits from them despite the fact that natural images display little resemblance with range-projection images. Second, we further compensate for ViTs' lack of inductive bias by substituting the classical linear embedding layer with a multi-layer convolutional stem. Finally, we refine pixel-wise predictions with a convolutional decoder and a skip connection from the convolutional stem to combine low-level but fine-grain features of the convolutional stem with the high-level but coarse predictions of the ViT encoder.

In summary, our contributions are the following: **(1)** To the best of our knowledge, we are the first to exploit the strong representation learning capacity of vision transformers architectures for 3D semantic segmentation from LiDAR point clouds. By revisiting, in the context of our problem, the tokenization process of the ViT's encoder and adding a light-weight convolutional decoder for refining the coarse patch-wise ViT representations, we derive a simple but effective projection-based LiDAR segmentation approach, which we call RangeViT. **(2)** Furthermore, as shown in Fig. 1, the proposed approach allows one to harness ViT models pre-trained on the RGB image domain for the LiDAR segmentation problem. Indeed, despite the large gap between the two domains, we empirically demonstrate that using such pre-training strategies improves segmentation performance. **(3)** Finally, our RangeViT approach, despite its simplicity, achieves state-of-the-art results among project-based segmentation methods.

2. Related work

2.1. CNNs for Point Cloud Segmentation

2D Methods Several works [3, 11, 24, 26, 28, 36, 42, 47, 50, 60, 64, 66, 70] project the 3D point cloud into the 2D space with range, perspective or bird's-eye-view (BEV) projection and process the projected images with 2D CNNs. For instance, PolarNet [66] employs bird's-eye-view projection to polar coordinates and then processes the bird-eye-view features with a Ring CNN. DarkNetSeg [3], SalsaNext [11], KPRNet [26], RangeNet++ [36], Lite-HDSeg [42] and SqueezeSegV3 [60] use range projection and then process the input images with a U-Net-like architecture. PMF [70], a multi-modal segmentation approach for point clouds and RGB images [2, 29, 53, 65, 70], projects the 3D points onto 2D camera frames and processes them together with the RGB images using a dual-stream CNN network.

3D Methods Instead of the 2D space, voxel-based approaches [20, 22, 35, 46] process the point clouds in their 3D space by first dividing the 3D space into voxels using cartesian coordinates and then applying 3D convolutions. Cylinder3D [69] show that dividing the 3D space into voxels using cylindrical coordinates instead of cartesian improves the segmentation performance. Although voxel-based methods consider the geometric properties of the 3D space, their drawback is that they are computationally expensive.

Also, for indoor-scene point cloud segmentation there are many methods [15, 37, 47, 52, 54, 56, 58, 62] relying on PointNet-inspired architectures [38] that directly process raw 3D points. However, these approaches cannot be easily adapted to outdoor-scene point cloud segmentation, which we consider in this work, as the large quantity of points cause computational difficulties.

2.2. ViTs for Point Cloud Segmentation

The recent rise to prominence of ViT architectures in computer vision has inspired a series of works for indoor point cloud segmentation [31, 57, 63, 67, 68].

Processing point clouds with a ViT model can be challenging and computationally expensive due to the large number of points. Point-BERT [63] uses farthest point sampling [39] and the K-NN algorithm to define input tokens for the transformer. Also, it proposes a self-supervised pre-training strategy for point data inspired by the masked token modeling approaches in the RGB image [23] and text [13] domains. Point Transformer [67] has a U-Net-like architecture but without convolutional layers and it integrates self-attention mechanism in all of its blocks. Finally, Bridged Transformer [57] jointly processes point clouds and RGB images with a fully transformer-based architecture.

Using a transformer-based architecture for the semantic segmentation task on outdoor LiDAR point clouds remains challenging. To the best of our knowledge, there has been no work published yet about outdoor LiDAR point cloud semantic segmentation with a ViT-based architecture.

2.3. Transfer Learning from Images to Point Clouds

ViT models have the capacity to learn powerful representations, but require vast amounts of training data for it. However, large LiDAR point cloud datasets are less common due to the costly and time consuming annotation procedure. Recent works [21,32,40,43,55,61] explore transfer learning of 2D image pre-trained models to 3D point clouds.

Image2Point [61] converts a 2D CNN into a 3D sparse CNN [61] by inflating 2D convolutions into 3D convolutions, which is done by repeating the 2D kernel along the third dimension. SLidR [43] is a self-supervised pre-training method on LiDAR data. It uses a student-teacher architecture, where the 2D teacher network pre-trained on images transfers information into the 3D student network. Pix4Point [40] and Simple3D-Former [55] study transfer learning from images to indoor point clouds with fully transformer-based architectures. They adapt the tokenizer and the head layer to be specialized for 3D point cloud data. Pix4Point [40] applies farthest point sampling [39], the K-NN algorithm and then a graph convolution on the aggregated neighborhoods to extract input tokens for the ViT encoder. Simple3D-Former [55] memorizes the ImageNet [12] representation of 2D image classification by incorporating a KL divergence term in the loss between its 2D image classification predictions and those obtained from a fixed ImageNet [12] pre-trained ViT network.

3. RangeViT

In this section, we describe our ViT-based LiDAR semantic segmentation approach, for which we provide an overview in Fig. 2.

3.1. General architecture

We represent a LiDAR point cloud of N points with a matrix $\mathbf{P} \in \mathbb{R}^{N\times 4}$. Each point $p = (x, y, z, i) \in \mathbf{P}$ has Cartesian coordinates denoted by (x, y, z) and LiDAR intensity denoted by i, and is annotated with a label $\ell \in \{1, \dots, K\}$ encoding one of the K semantic classes.

Range projection. The input of our RangeViT backbone is a 2D representation of the input point cloud. We use the well-known range projection [36]. Each point $\mathbf{p} \in \mathbf{P}$ with coordinates (x, y, z) is projected on a range image of size $H \times W$. The projected 2D coordinates satisfies

$$\begin{pmatrix} h \\ w \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\left(1 - \arctan(y, x)\pi^{-1}\right) W \\ \left(1 - \left(\arcsin(z, r^{-1}) + |f_{\text{down}}|\right) f_v^{-1}\right) H \end{pmatrix}, \quad (1)$$

where $f_v = |f_{\text{down}}| + |f_{\text{up}}|$ is the vertical field-of-view of the LiDAR sensor. We associate C=5 low-level features (r, x, y, z, i) to each projected point, where $r = \sqrt{x^2 + y^2 + z^2}$ is the range of the corresponding point (i.e., its distance from the LiDAR sensor), to create the range image $\mathbf{I} \in \mathbb{R}^{C\times H\times W}$. Note that if more than one point is projected onto the same pixel, then only the feature with the smallest range is kept. Pixels with no point projected on them have their features filled with zeros.

Convolutional stem. In a standard ViT, the image is divided into M patches of size $P_H \times P_W$, which are linearly embedded to provide D-dimensional visual tokens. Yet, our empirical study in Sec. 4.2 shows that this tokenization process of standard ViTs is far from optimal on both our task and datasets of interest. In order to bridge this potential domain gap between range images and standard ViTs, we replace the embedding layer with a non-linear convolutional stem [59]. Non-linear convolutional stems have been shown to increase optimization stability and predictive performance of ViTs [59], whereas we leverage them primarily for *steering* range images towards ViT-like inputs.

The first part of the convolutional stem consists of the first 4 residual blocks of SalsaNext [11], called context module. This context module captures short-range dependencies of the 3D points projected in the range image and produces pixel-wise features with D_{h} channels at the same resolution of the input range image, hence the tensor of context features $\mathbf{t}_{\text{c}}$ has size $D_{\text{h}} \times H \times W$.[1] Then, in order to produce tokens compatible with the input of a ViT, we use an average pooling layer that reduces the spatial dimensions of the context features $\mathbf{t}_{\text{c}}$ from $H \times W$ to $(H/P_H) \times (W/P_W)$, and use a final 1×1 convolutional layer with D output channels. The convolutional stem thus yields $M = (HW)/(P_H P_W)$ visual tokens $\mathbf{v}_1, \dots, \mathbf{v}_M$ of dimension D, i.e., matching the input dimension and number of tokens of a traditional ViT.

[1]The first 3 residual layers actually have $D_{\text{in}} = 32$ channels, which is typically smaller than D_{h}.

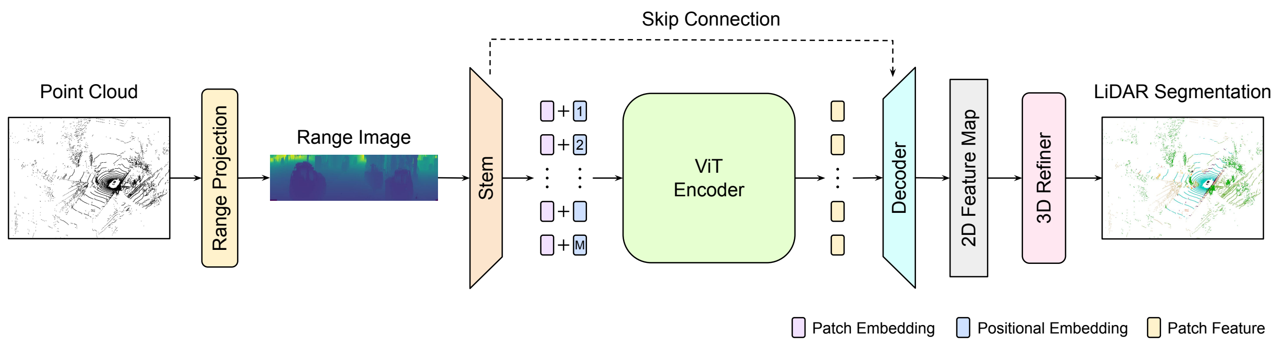

Figure 2. **Overview of RangeViT architecture.** First, the point cloud is projected in a 2D space with range projection. Then, the produced range image is processed by the convolutional stem, the ViT encoder and the decoder to obtain a 2D feature map. It is then processed by a 3D refiner layer for 3D point-wise predictions. Note that there is a single skip connection between the convolutional stem and the decoder.

ViT encoder. The output of the convolutional stem can be fed directly to a ViT [14]. The input $\mathbf{t}_0$ to our ViT encoder is prepared by stacking all the visual tokens $\mathbf{v}_1, \ldots, \mathbf{v}_M$ and a classification token $\mathbf{v}_{\text{class}} \in \mathbb{R}^D$, to which we add the positional embeddings $\mathbf{E}_{\text{pos}} \in \mathbb{R}^{(M+1)\times D}$:

$$\mathbf{t}_0 = [\mathbf{v}_{\text{class}}, \mathbf{v}_1, \ldots, \mathbf{v}_M] + \mathbf{E}_{\text{pos}}. \quad (2)$$

The input tokens are then transformed by the ViT encoder to obtain an updated sequence of tokens $\mathbf{t}_L \in \mathbb{R}^{(M+1)\times D}$, where L denotes the number of transformer blocks. Then, we remove the classification token from $\mathbf{t}_L$ to keep only the deep patch representations $\mathbf{t}'_L \in \mathbb{R}^{M \times D}$.

Decoder. The representations $\mathbf{t}'_L$ provided by the ViT encoder are patch representations which are unfortunately too coarse to obtain good point predictions. Therefore, we use a decoder to refine these coarse patch representations. First, $\mathbf{t}'_L$ is reshaped in form of a 2D feature map of size $D \times H/P_H \times W/P_W$. Our convolutional decoder consists of a $1{\times}1$ convolution layer with $D_{\text{h}} P_H P_W$ output channels, followed by a Pixel Shuffle layer [44] which yields feature maps with shape $D_{\text{h}} \times H \times W$, i.e., with the same resolution as the original range image. While the convolutional decoder can still produce coarse features or decoding artifacts, the Pixel Shuffle is particularly effective in recovering fine information from features. Then, we concatenate these features with the context features $\mathbf{t}_{\text{c}}$ from the convolutional stem and use a series of two convolutional layers with $3{\times}3$ and $1{\times}1$ kernels respectively, each of them followed by Leaky ReLU and batch normalisation, to obtain the refined feature map $\mathbf{t}_{\text{dec}} \in \mathbb{R}^{D_{\text{h}} \times H \times W}$.

3D refiner. Ultimately, we need to convert the pixel-wise features from the range image space into point-wise predictions in the 3D space. Most prior range-projection based methods first make pixel-wise class predictions and then unproject them to the 3D space, where at inference time often there is a post-processing step relying, e.g., on K-NN [36] or CRFs [27]. The purpose of the latter post-processing step is to fix segmentation mistakes related to the projection and processing of 3D points in a 2D space (e.g., multiple 3D points being projected on the same pixel, or 2D boundary prediction errors for points that are actually far away in the 3D space). Instead, we follow the approach of KPRNet [26] that proposes an end-to-end approach that learns this post-processing step with a KPConv [47] layer.

KPConv [47] is a point convolution technique which works directly on the original 3D points. It permits to leverage the underlying geometry of the 3D point clouds to refine features at the point level. So, in our network, we project the 2D feature maps $\mathbf{t}_{\text{dec}}$ of the 2D decoder back to original 3D points by bilinear upsampling, thus obtaining point-wise features with shape $N \times D_{\text{h}}$, where N is the number of 3D points. Then, these point features are given to the KPConv layer as input along with the 3D coordinates of the corresponding points, which outputs D_{h}-dimensional point features. Finally, the logits $\mathbf{s} \in \mathbb{R}^{N \times K}$ are obtained by applying a BatchNorm, a ReLU and a final point-wise linear layer on these point features.

3.2. Implementation details

Training loss We use the sum of the multi-class focal loss [30] and the Lovász-softmax loss [4]. The focal loss is a scaled version of the cross-entropy loss [19] adapting its penalty to the hardness of the samples, making it suited for datasets with class imbalance, such as semantic segmentation. The Lovász-softmax is developed specifically for semantic segmentation and built to optimize the mIoU.

Inference As in [45], we use a sliding-window method during inference. The network actually never sees the entire range images during training but only crops extracted from it. At inference, the range image is divided into overlapping crops of the same size as those used during training. The corresponding 2D features at the output of the decoder are

Stem	Decoder	Refiner	mIoU (%)	#Params
Linear	Linear		65.52	22.0M
Conv	Linear		69.82	22.8M
Conv	UpConv		73.83	24.6M
Conv	UpConv	✓	**74.60**	25.2M

Table 1. **Model ablations.** Results on the nuScenes validation set with $D_{\rm h} = 192$. The linear stem refers to the linear patch embedding layer. When the 3D refiner layer (Refiner column) is not used, we use the K-NN post-processing technique [36].

then averaged to reconstruct the entire feature map, which is then processed by our 3D refinement layer.

4. Experiments

4.1. Experimental Setup

Datasets and metrics. We validate our approach for 3D point cloud semantic segmentation on two different commonly used datasets: nuScenes [7] and SemanticKITTI [3]. We conduct most of our ablation studies on nuScenes and compare against previous works on both nuScenes and SemanticKITTI. As evaluation metric, we use the mean Intersection over Union (mIoU) [16].

NuScenes [7] consists of 1,000 scenes of 20 seconds in Boston and Singapore with various urban scenarios, lighting and weather conditions. The LiDAR sensor has 32 laser beams. Furthermore, there are 16 annotated semantic classes and the dataset is split into 28,130 training and 6,019 validation point cloud scans.

SemanticKITTI [3] is created from the KITTI Vision Odometry Benchmark [17] and consists of urban scenes collected in Germany. Sequences 00-10 are used for training, except sequence 08 which is used for validation. There are 19,130 training and 4,071 validation scans. Sequences 11-21 are used for test and they contain 20,351 scans. The LiDAR sensor has 64 laser beams and there are 19 annotated semantic classes.

Model and pre-trained weights. For all experiments, we use the ViT-S/16 model [14] as the encoder. It has L=12 layers, 6 attention heads and D=384 channels, amounting to approximately 21M parameters. Unless otherwise stated, (a) this ViT encoder is initialized with weights pre-trained on ImageNet21k [12] for classification and then fine-tuned on Cityscapes [10] for semantic image segmentation [45]; (b) the stem, decoder and 3D refiner layers, which are always randomly initialized, use $D_{\rm h} = 256$ feature channels.

Optimization. We use the AdamW optimizer [34] with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and weight decay 0.01. The batch-size is 32 and 16 for nuScenes and SemanticKITTI, respectively. For the learning rate lr, we use a linear warm-up from 0 to its peak value for 10 epochs and then we decrease it over the remaining training epochs to 0 with a cosine annealing schedule [33]. In SemanticKITTI, we use 60 training epochs and the peak lr is 4×10^{-4}. In nuScenes, we use 150 epochs and the peak lr is 8×10^{-4} when training from Cityscapes, ImageNet21k and random initializations, and 2×10^{-4} when training from DINO initialization.

$D_{\rm h}$ size	64	128	192	256
#Params	22.7M	23.7M	25.2M	27.1M
nuScenes mIoU (%)	74.00	74.12	74.60	**75.21**
SemanticKITTI mIoU (%)	58.48	**60.73**	60.45	60.55

Table 2. **Impact of the channel size** $D_{\rm h}$ used in the stem, decoder, and 3D refiner layers. Results on the nuScenes and the SemanticKITTI validation sets.

Data augmentations. As point cloud augmentations we use: (a) flips over the y axis (the vertical axis on the range image), (b) random translations, and (c) random rotations between $\pm 5^\circ$ (using the roll, pitch and yaw angles). All augmentations are applied randomly with probability 0.5. Finally, after range projection, we take a random image crop with a fixed size of 32×384 for nuScenes and 64×384 for SemanticKITTI. Note that the full size of a range image is 32×2048 for nuScenes and 64×2048 for SemanticKITTI.

4.2. What makes a ViT architecture for 3D semantic segmentation?

Our aim is to adapt to LiDAR point clouds with the fewest possible modifications of the standard ViT architecture [14], which already follows faithfully the original self-attention Transformer design [51]. Differently from ViT that performs classification on images, RangeViT performs semantic segmentation of point cloud-derived range images. We study here the possible choices for the input processing and decoder layers using the nuScenes dataset.

Stem and decoder. In Tab. 1, we study the impact of the convolutional stem and the UpConv decoder. To that end, we report results using a linear patch embedding [14] as a stem and a linear decoder.[2] Starting from a model with linear stem and linear decoder, introducing the proposed convolutional stem leads to a significant mIoU boost ($65.52 \rightarrow 69.82$). This highlights the importance of having a non-linear network component for producing appropriate input token features for the ViT backbone. In addition, when using image pre-trained ViTs (Sec. 4.3) the convolutional stem can effectively steer the distribution of input range images towards the image-based feature distribution the ViT has been pre-trained on, leading to smoother fine-tuning. Then, also replacing the linear decoder with the proposed UpConv decoder leads to another notable improvement on mIoU ($69.8 \rightarrow 73.83$), validating this design choice. Finally, replacing the K-NN refinement [36] with

[2]The linear decoder consists of a simple 1×1 conv. layer, which predicts patch-wise classification scores, and a bilinear upsampling to reach the input image resolution.

Patch size	16 × 16	8 × 8	4 × 16	4 × 8	4 × 4	2 × 16	2 × 8
mIoU (%)	68.45	72.04	72.72	73.30	73.70	73.88	**75.21**
#Tokens	49	193	193	385	769	385	769
Train time	×1	×1.02	×1.02	×1.13	×1.43	×1.13	×1.43

Table 3. **Patch size ablations.** Results on the nuScenes validation set. The mIoU scores are in increasing order. The train time is relative to the 16×16 patch size case.

the KPConv layer (in the 3D refiner layer), leads to a small but non-trivial mIoU increase (73.83 → 74.60).

In Tab. 2, we study what is the impact, on the mIoU, of changing the number of feature channels D_{h} on the stem, decoder and 3D refiner layer. On nuScenes, we observe that the results gradually increase when increasing the dimension D_{h}. In the SemanticKITTI case, $D_{\mathrm{h}} = 128$ gives the best results and further increasing D_{h} does not help. Note that the size of the ViT's embeddings is fixed at $D = 384$, regardless of the D_{h} size.

What patch-size for range image "tokenization"? The size of the input patch tokens is an essential factor for controlling the trade-off between speed and accuracy performance without changing the number of parameters in the model. Intuitively, reducing the patch-size results in a finer representation, but also a longer input sequence that takes longer to process. Conversely, larger patches lead to a coarser representation, yet faster inference. The performance impact of the patch-size can sometimes match the one of model size [45]. For simplicity, standard ViT models use square patches of size 32, 16 or 8, as they are typically trained on square images. In contrast, range images have a much different aspect ratio (high width) and different layout of the content in the image (rows of points corresponding to LiDAR beams). With this insight, we revisit the practice of using square patches and look into rectangular patches with different aspect ratio (1:2, 1:4, 1:8) that would better capture the specific local patterns of the range images.

Tab. 3 shows the mIoU scores for different patch sizes for dividing the range image to patch tokens. Note that the convolutional stem first produces pixel-wise features and then, given a patch-size (P_H, P_W), reduces the spatial dimensions of these features to $(H/P_H, W/P_W)$ using local average pooling. We observe that smaller patch area is better but not necessarily with a square patch, commonly used for ViT models. The wide range images benefit more from rectangular patches (2×8 performing best). The smaller patches enable the extraction of more fine-grained information leading to more precise predictions for smaller and thinner objects and points at object boundaries.

4.3. Exploiting image pre-trained ViTs

So far, we have studied the architectural choices necessary for using ViT models for point cloud semantic segmentation. The final RangeViT model preserves the ViT backbone intact allowing us to initialize it with weights from models pre-trained on large datasets. We study now whether using such an initialization helps or not and what fine-tuning strategies would be more suitable in this context.

Pre-training	Rand	DINO	IN21k	CS
mIoU (%)	72.37	73.33	74.77	**75.21**

Table 4. **ViT pre-training on RGB images.** Comparison of different weight initializations of the ViT encoder on the nuScenes validation set. Rand: randomly initialized. The positional embeddings are initialized with the corresponding pre-trained weights or randomly when training from scratch. The convolutional stem, the decoder and the 3D refiner layer are randomly initialized.

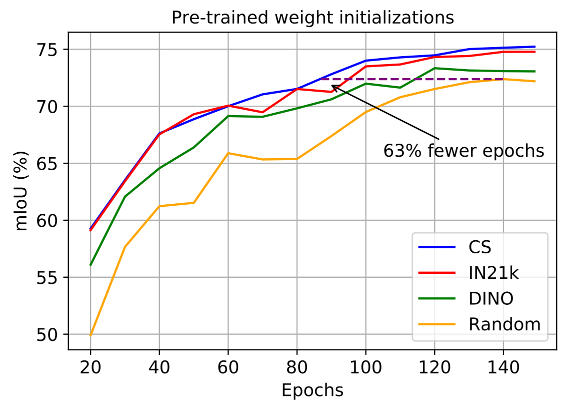

Figure 3. **Training efficiency with image-pretrained ViTs.** Comparison of the validation mIoU curves for different weight initialisations of the ViT encoder on the nuScenes validation set.

Is pre-training on RGB images beneficial? In Tab. 4, we study the effect of transferring to our task ViT models pre-trained on natural RGB images. In particular, we explore initializing RangeViT's backbone with ViTs pre-trained: (a) on supervised ImageNet21k classification [14] (entry IN21k), (b) on supervised image segmentation on Cityscapes with Segmenter [45] (entry CS), which in its turn was pre-trained with IN21k, and (c) with the DINO [9] self-supervised approach on ImageNet1k (entry DINO).

We observe that, despite the large domain gap, using ViT models pre-trained on RGB images is always better than training from scratch on LiDAR data (entry Rand). For instance, using the IN21k and CS pretrained ViTs leads to improving the mIoU scores by 2.4 and 2.8 points respectively. Additionally, as we see in Fig. 3, which plots the nuScenes validation mIoU as a function of the training epochs, using such pre-trained ViTs leads to faster training convergence of the LiDAR segmentation model. We argue that this is a highly interesting finding. It means that our RangeViT approach, by being able to use off-the-shelf pre-trained ViT models, can directly benefit from current and future advances on the training ViT models with natural RGB images, a very active and rapidly growing re-

Model	Fine-tuning LN	ATTN	FFN	IN21k mIoU (%)	CS mIoU (%)
(a)	✓	✓	✓	74.79	75.21
(b)				67.88	68.03
(c)	✓			69.08	69.31
(d)	✓	✓		73.56	72.77
(e)	✓		✓	75.11	75.47

Table 5. **Partial ViT fine-tuning.** We use a ViT encoder pre-trained on ImageNet21k (IN21k) or Cityscapes (CS) and partially fine-tune RangeViT on the nuScenes training set. The convolutional stem, the positional embeddings and the UpConv decoder are always fine-tuned. LN: fine-tuning the LayerNorm layers of the ViT encoder. ATTN: fine-tuning the multi-head attention layers of the ViT encoder. FFN: fine-tuning the feed-forward network layers of the ViT encoder. Model (a) is full fine-tuning of the network. The results are reported on the nuScenes validation set.

Encoder	ViT-S†	ViT-S	RN50†	RN50	Identity
mIoU (%)	67.88	**74.77**	60.48	72.30	53.73

Table 6. **Ablating encoder backbones.** ViT-S and RN50 are pre-trained on IN21k. †: the encoder remains frozen during training.

search field [23, 41, 48, 49]. Furthermore, from the small difference between the mIoU scores with the IN21k and CS pre-trainings, we infer that the pre-training can lead to consistent performance improvements even if it is not on the strongly-supervised image segmentation task, which requires expensive-to-annotate datasets.

Which ViT layers is better to fine-tune? In the image domain, several practices have emerged for fine-tuning a pre-trained convolutional network on a downstream dataset. Taking into consideration the domain gap between the pre-training and downstream data and the amount of labeled data available, the entire network can be fine-tuned or only a part of it. Here the domain gap between RGB images and range images is major and we would expect a full fine-tuning of the network to be better. To understand how much prior knowledge of the image pre-trained ViT is useful for point clouds, we study different fine-tuning strategies: fine-tuning all layers, only the attention (ATTN) layers or only the feedforward network (FFN) layers.

In Tab. 5, we show results with these different fine-tuning strategies. Interestingly, the best results are not achieved with full fine-tuning (model (a)) but when the attention layers are kept frozen (model (e)). This suggests that the pre-trained ATTN layers are already well learnt and ready to generalize to range images. With CS pre-training ATTN layers may capture the layout of the scenes more easily without much fine-tuning, as the LiDAR scans have been also acquired from urban road scenes. Fine-tuning FFN may have more impact due to the different specifics of the LiDAR data compared to RGB images. In addition, FFN layers are in practice easier and more stable to optimize

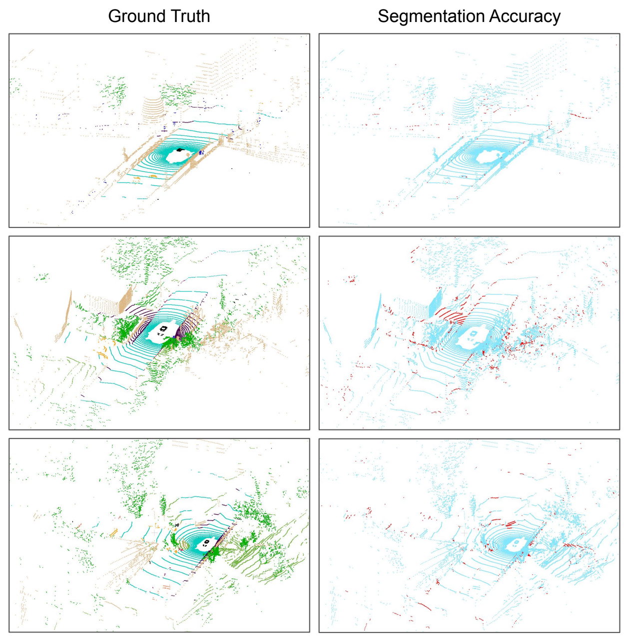

Figure 4. **Visualizing the segmentation accuracy.** Left: ground truth segmentation of validation point clouds of nuScenes, coloured based on their label. Right: segmentation accuracy of RangeViT – good predictions are in blue and the bad ones in red.

(they are essentially fully-connected and normalization layers) than ATTN layers that usually require more careful hyper-parameter selection. These findings align with recent ones from the image domain [48], confirming that the convolutional stems do steer the input LiDAR data to behave like image data once on in the ViT backbone.

Ablating encoder backbones. In Tab. 6, we replace the ViT-S encoder backbone of RangeViT with a ResNet-50 (RN50) encoder or the identity function (i.e., the decoder follows directly the stem). Both ViT-S and RN50 are pre-trained on IN21k. We see that switching from ViT-S to RN50 decreases the mIoU from 74.77% to 72.30%. Furthermore, when the backbones remain frozen (ViT-S† and RN50† models), we reach 67.88% with ViT-S and 60.48% with RN50, demonstrating that ViT features are more appropriate for transfer learning to LiDAR data than CNNs. Finally, with the Identity backbone, we achieve 53.73%, which is more than 14 points worse than the ViT-S† model that has the same number of learnable parameters.

4.4. Comparison to the state of the art

Tab. 7 and 8 report the final comparison on the nuScenes validation set and on the SemanticKITTI test set including class-wise IoU scores. We observe that our model achieves superior mIoU performance compared to prior 2D-projection based methods on both datasets, reducing the gap with the strong voxel-based Cylinder3D [69] method.

Method	barrier	bicycle	bus	car	construction	motorcycle	pedestrian	traffic cone	trailer	truck	driveable	other flat	sidewalk	terrain	manmade	vegetation	**mIoU (%)**
Voxel-based																	
Cylinder3D [69]	**76.4**	40.3	**91.3**	**93.8**	**51.3**	78.0	**78.9**	64.9	62.1	**84.4**	**96.8**	**71.6**	**76.4**	**75.4**	**90.5**	**87.4**	**76.1**
2D Projection-based																	
RangeNet++ [36]	66.0	21.3	77.2	80.9	30.2	66.8	69.6	52.1	54.2	72.3	94.1	66.6	63.5	70.1	83.1	79.8	65.5
PolarNet [66]	74.7	28.2	85.3	90.9	35.1	77.5	71.3	58.8	57.4	76.1	96.5	71.1	74.7	74.0	87.3	85.7	71.0
SalsaNext [11]	74.8	34.1	85.9	88.4	42.2	72.4	72.2	63.1	61.3	76.5	96.0	70.8	71.2	71.5	86.7	84.4	72.2
RangeViT-IN21k (ours)	75.1	39.0	90.2	88.4	48.0	79.2	77.2	**66.4**	65.1	76.7	96.3	71.1	73.7	73.9	88.9	87.1	74.8
RangeViT-CS (ours)	75.5	**40.7**	88.3	90.1	49.3	**79.3**	77.2	66.3	**65.2**	80.0	96.4	71.4	73.8	73.8	89.9	87.2	75.2

Table 7. **nuScenes validation set comparison** with state-of-the-art methods. The best results are bold and the second best results are blue.

Method	car	bicycle	motorcycle	truck	other vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	other ground	building	fence	vegetation	trunk	terrain	pole	traffic sign	**mIoU (%)**
Voxel-based																				
Cylinder3D [69]	**97.1**	**67.6**	**64.0**	**59.0**	**58.6**	**73.9**	67.9	36.0	91.4	65.1	75.5	32.3	91.0	66.5	85.4	**71.8**	68.5	**62.6**	65.6	**67.8**
2D Projection-based																				
RangeNet++ [36]	91.4	25.7	34.4	25.7	23.0	38.3	38.8	4.8	91.8	65.0	75.2	27.8	87.4	58.6	80.5	55.1	64.6	47.9	55.9	52.2
PolarNet [66]	93.8	40.3	30.1	22.9	28.5	43.2	40.2	5.6	90.8	61.7	74.4	21.7	90.0	61.3	84.0	65.5	67.8	51.8	57.5	54.3
SqueezeSegV3 [60]	92.5	38.7	36.5	29.6	33.0	45.6	46.2	20.1	91.7	63.4	74.8	26.4	89.0	59.4	82.0	58.7	65.4	49.6	58.9	55.9
SalsaNext [11]	91.9	48.3	38.6	38.9	31.9	60.2	59.0	19.4	91.7	63.7	75.8	29.1	90.2	64.2	81.8	63.6	66.5	54.3	62.1	59.5
KPRNet [26]	95.5	54.1	47.9	23.6	42.6	65.9	65.0	16.5	**93.2**	**73.9**	**80.6**	30.2	91.7	68.4	**85.7**	69.8	**71.2**	58.7	64.1	63.1
Lite-HDSeg [42]	92.3	40.0	55.4	37.7	39.6	59.2	**71.6**	**54.1**	93.0	68.2	78.3	29.3	91.5	65.0	78.2	65.8	65.1	59.5	**67.7**	63.8
RangeViT-CS (ours)	95.4	55.8	43.5	29.8	42.1	63.9	58.2	38.1	93.1	70.2	80.0	**32.5**	**92.0**	**69.0**	85.3	70.6	**71.2**	60.8	64.7	64.0

Table 8. **SemanticKITTI test set comparison** with state-of-the-art methods. The best results are bold and the second best results are blue.

Class-wise IoU result analysis. In Tab. 7 and 8, we can see that RangeViT often achives the best or the second best class-wise IoU scores. In both datasets, the classes are imbalanced and due to the sparsity and varying density of LiDAR point clouds, some classes (e.g., bicycle, pedestrian) are represented with few, not necessarily structured points per scene, so it is difficult recognize them. This problem could possibly be reduced by jointly processing point clouds and RGB images, since RGB images might provide extra cues about the outline and the shape of these objects.

4.5. Qualitative Results

We visualize the 3D point clouds with the ground truth semantic labels as well as the predictions with CloudCompare [18]. Fig. 4 shows the predictions of RangeViT on three validation point clouds of nuScenes. We notice minor errors such as man-made objects predicted as sidewalk and imperfect borders for the vegetation. We also remark difficulties in recognizing pedestrians and confusion between the driveable surface and the sidewalk for few points.

5. Conclusion

We studied the feasibility of leveraging (pre-trained) ViTs for LiDAR 3D semantic segmentation with projection-based methods. We discover that in spite of the significant domain gap between RGB images and range images and their high requirements of training data, ViTs can be successfully used without any changes in the original transformer backbone. We achieve this thanks to an adapted tokenization and pre-processing for the ViT encoder and a simple convolutional decoder. We show that ViTs pre-trained on large image datasets can be effectively repurposed for LiDAR segmentation towards reaching state-of-the-art performance among 2D projection methods. We release the code for our implementation and hope that it could be used as a testbed for evaluating the ability of ViT image "foundation" models [6] to generalize on different domains.

Future work. Although the results are promising, there is still room for improvement. For instance, we identified the tokenization of LiDAR data as a crucial factor for success. As future work, we could further improve this process, e.g., with FlexiViT [5] (random patch sizes) or Perceiver IO [25] (learning to extract tokens), and consider tokenizing raw 3D data instead of the 2D projections.

Acknowledgements. We would like to express our gratitude to Matthieu Cord and Hugo Touvron for their insightful comments, Robin Strudel *et al*. [45] for kindly providing ViT-S pre-trained on Cityscapes, and Oriane Siméoni for her valuable contribution to certain experiments. This work was performed using HPC resources from GENCI-IDRIS (Grants 2022-AD011012884R1, 2022-AD011013413 and 2022-AD011013701). The authors acknowledge the support of the French Agence Nationale de la Recherche (ANR), under grant ANR-21-CE23-0032 (project MultiTrans).

References

[1] Eren Erdal Aksoy, Saimir Baci, and Selcuk Cavdar. SalsaNet: Fast road and vehicle segmentation in LiDAR point clouds for autonomous driving. In *IV*, 2020. 1

[2] Xuyang Bai, Zeyu Hu, Xinge Zhu, Qingqiu Huang, Yilun Chen, Hongbo Fu, and Chiew-Lan Tai. TransFusion: Robust LiDAR-camera fusion for 3D object detection with transformers. In *CVPR*, 2022. 2

[3] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences. In *ICCV*, 2019. 2, 5

[4] Maxim Berman, Amal Rannen Triki, and Matthew B Blaschko. The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In *CVPR*, 2018. 4

[5] Lucas Beyer, Pavel Izmailov, Alexander Kolesnikov, Mathilde Caron, Simon Kornblith, Xiaohua Zhai, Matthias Minderer, Michael Tschannen, Ibrahim Alabdulmohsin, and Filip Pavetic. FlexiViT: One model for all patch sizes. In *CVPR*, 2023. 8

[6] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. In *arXiv*, 2021. 8

[7] 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 *CVPR*, 2020. 5

[8] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 2

[9] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021. 2, 6

[10] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 5

[11] Tiago Cortinhal, George Tzelepis, and Eren Erdal Aksoy. SalsaNext: Fast, uncertainty-aware semantic segmentation of LiDAR point clouds. In *ISVC*, 2020. 1, 2, 3, 8

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009. 3, 5

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *arXiv*, 2018. 3

[14] 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. In *ICLR*, 2021. 1, 2, 4, 5, 6

[15] Francis Engelmann, Martin Bokeloh, Alireza Fathi, Bastian Leibe, and Matthias Nießner. 3D-MPA: Multi-proposal aggregation for 3D semantic instance segmentation. In *CVPR*, 2020. 2

[16] Mark Everingham, SM Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. In *IJCV*, 2015. 5

[17] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *CVPR*, 2012. 5

[18] Daniel Girardeau-Montaut. CloudCompare. *France: EDF R&D Telecom ParisTech*, 2016. 8

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016. 4

[20] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3D semantic segmentation with submanifold sparse convolutional networks. In *CVPR*, 2018. 2

[21] Saurabh Gupta, Judy Hoffman, and Jitendra Malik. Cross modal distillation for supervision transfer. In *CVPR*, 2016. 3

[22] Lei Han, Tian Zheng, Lan Xu, and Lu Fang. OccuSeg: Occupancy-aware 3D instance segmentation. In *CVPR*, 2020. 2

[23] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022. 3, 7

[24] 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. In *CVPR*, 2020. 2

[25] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver IO: A general architecture for structured inputs & outputs. In *ICLR*, 2022. 8

[26] Deyvid Kochanov, Fatemeh Karimi Nejadasl, and Olaf Booij. KPRNet: Improving projection-based LiDAR semantic segmentation. In *ECCV*, 2020. 1, 2, 4, 8

[27] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected CRFs with gaussian edge potentials. In *NeurIPS*, 2011. 4

[28] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *CVPR*, 2018. 2

[29] Yingwei Li, Adams Wei Yu, Tianjian Meng, Ben Caine, Jiquan Ngiam, Daiyi Peng, Junyang Shen, Yifeng Lu, Denny Zhou, Quoc V Le, et al. DeepFusion: LiDAR-camera deep fusion for multi-modal 3D object detection. In *CVPR*, 2022. 2

[30] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 4

[31] Haotian Liu, Mu Cai, and Yong Jae Lee. Masked discrimination for self-supervised learning on point clouds. In *ECCV*, 2022. 2

[32] Yueh-Cheng Liu, Yu-Kai Huang, Hung-Yueh Chiang, Hung-Ting Su, Zhe-Yu Liu, Chin-Tang Chen, Ching-Yu Tseng, and

Winston H Hsu. Learning from 2D: Contrastive pixel-to-point knowledge transfer for 3D pretraining. In *arXiv*, 2021. 3
[33] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *ICLR*, 2017. 5
[34] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 5
[35] Hsien-Yu Meng, Lin Gao, Yu-Kun Lai, and Dinesh Manocha. VV-Net: Voxel VAE net with group convolutions for point cloud segmentation. In *ICCV*, 2019. 2
[36] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. RangeNet++: Fast and accurate LiDAR semantic segmentation. In *IROS*, 2019. 1, 2, 3, 4, 5, 8
[37] Quang-Hieu Pham, Thanh Nguyen, Binh-Son Hua, Gemma Roig, and Sai-Kit Yeung. JSIS3D: Joint semantic-instance segmentation of 3D point clouds with multi-task pointwise networks and multi-value conditional random fields. In *CVPR*, 2019. 2
[38] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *CVPR*, 2017. 2
[39] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017. 3
[40] Guocheng Qian, Xingdi Zhang, Abdullah Hamdi, and Bernard Ghanem. Pix4Point: Image pretrained transformers for 3D point cloud understanding. In *arXiv*, 2022. 3
[41] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 7
[42] Ryan Razani, Ran Cheng, Ehsan Taghavi, and Liu Bingbing. Lite-HDSeg: LiDAR semantic segmentation using lite harmonic dense convolutions. In *ICRA*, 2021. 2, 8
[43] Corentin Sautier, Gilles Puy, Spyros Gidaris, Alexandre Boulch, Andrei Bursuc, and Renaud Marlet. Image-to-LiDAR self-supervised distillation for autonomous driving data. In *CVPR*, 2022. 3
[44] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*, 2016. 4
[45] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *ICCV*, 2021. 2, 4, 5, 6, 8
[46] Lyne Tchapmi, Christopher Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese. SEGCloud: Semantic segmentation of 3D point clouds. In *3DV*, 2017. 2
[47] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. KPConv: Flexible and deformable convolution for point clouds. In *CVPR*, 2019. 2, 4
[48] Hugo Touvron, Matthieu Cord, Alaaeldin El-Nouby, Jakob Verbeek, and Hervé Jégou. Three things everyone should know about vision transformers. In *ECCV*, 2022. 7
[49] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *ICCV*, 2021. 7
[50] Larissa T Triess, David Peter, Christoph B Rist, and J Marius Zöllner. Scan-based semantic segmentation of lidar point clouds: An experimental study. In *IV*, 2020. 2
[51] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 1, 5
[52] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018. 2
[53] Chunwei Wang, Chao Ma, Ming Zhu, and Xiaokang Yang. PointAugmenting: Cross-modal augmentation for 3D object detection. In *CVPR*, 2021. 2
[54] Lei Wang, Yuchun Huang, Yaolin Hou, Shenman Zhang, and Jie Shan. Graph attention convolution for point cloud semantic segmentation. In *CVPR*, 2019. 2
[55] Yi Wang, Zhiwen Fan, Tianlong Chen, Hehe Fan, and Zhangyang Wang. Can we solve 3D vision tasks starting from a 2D vision transformer? In *arXiv*, 2022. 3
[56] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph CNN for learning on point clouds. In *TOG*, 2019. 2
[57] Yikai Wang, TengQi Ye, Lele Cao, Wenbing Huang, Fuchun Sun, Fengxiang He, and Dacheng Tao. Bridged Transformer for vision and point cloud 3D object detection. In *CVPR*, 2022. 2, 3
[58] Wenxuan Wu, Zhongang Qi, and Li Fuxin. PointConv: Deep convolutional networks on 3D point clouds. In *CVPR*, 2019. 2
[59] Tete Xiao, Mannat Singh, Eric Mintun, Trevor Darrell, Piotr Dollár, and Ross Girshick. Early convolutions help transformers see better. In *NeurIPS*, 2021. 3
[60] Chenfeng Xu, Bichen Wu, Zining Wang, Wei Zhan, Peter Vajda, Kurt Keutzer, and Masayoshi Tomizuka. SqueezeSegV3: Spatially-adaptive convolution for efficient point-cloud segmentation. In *ECCV*, 2020. 1, 2, 8
[61] Chenfeng Xu, Shijia Yang, Bohan Zhai, Bichen Wu, Xiangyu Yue, Wei Zhan, Peter Vajda, Kurt Keutzer, and Masayoshi Tomizuka. Image2Point: 3D point-cloud understanding with pretrained 2D convnets. In *arXiv*, 2021. 3
[62] Xu Yan, Chaoda Zheng, Zhen Li, Sheng Wang, and Shuguang Cui. PointASNL: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In *CVPR*, 2020. 2
[63] 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. In *CVPR*, 2022. 2, 3
[64] Feihu Zhang, Jin Fang, Benjamin Wah, and Philip Torr. Deep FusionNet for point cloud semantic segmentation. In *ECCV*, 2020. 2
[65] Yanan Zhang, Jiaxin Chen, and Di Huang. CAT-Det: Contrastively augmented transformer for multi-modal 3D object detection. In *CVPR*, 2022. 2

[66] Yang Zhang, Zixiang Zhou, Philip David, Xiangyu Yue, Zerong Xi, Boqing Gong, and Hassan Foroosh. PolarNet: An improved grid representation for online LiDAR point clouds semantic segmentation. In *CVPR*, 2020. 1, 2, 8

[67] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point Transformer. In *ICCV*, 2021. 2, 3

[68] Junsheng Zhou, Xin Wen, Yu-Shen Liu, Yi Fang, and Zhizhong Han. Self-supervised point cloud representation learning with occlusion auto-encoder. In *arXiv*, 2022. 2

[69] Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Wei Li, Yuexin Ma, Hongsheng Li, Ruigang Yang, and Dahua Lin. Cylindrical and asymmetrical 3D convolution networks for LiDAR-based perception. In *CVPR*, 2021. 2, 7, 8

[70] Zhuangwei Zhuang, Rong Li, Kui Jia, Qicheng Wang, Yuanqing Li, and Mingkui Tan. Perception-aware multi-sensor fusion for 3D LiDAR semantic segmentation. In *ICCV*, 2021. 2