MemorySeg: Online LiDAR Semantic Segmentation with a Latent Memory

Enxu Li Sergio Casas Raquel Urtasun
Waabi University of Toronto
{tli, sergio, urtasun}@waabi.ai

Abstract

Semantic segmentation of LiDAR point clouds has been widely studied in recent years, with most existing methods focusing on tackling this task using a single scan of the environment. However, leveraging the temporal stream of observations can provide very rich contextual information on regions of the scene with poor visibility (e.g., occlusions) or sparse observations (e.g., at long range), and can help reduce redundant computation frame after frame. In this paper, we tackle the challenge of exploiting the information from the past frames to improve the predictions of the current frame in an online fashion. To address this challenge, we propose a novel framework for semantic segmentation of a temporal sequence of LiDAR point clouds that utilizes a memory network to store, update and retrieve past information. Our framework also includes a novel regularizer that penalizes prediction variations in the neighborhood of the point cloud. Prior works have attempted to incorporate memory in range view representations for semantic segmentation, but these methods fail to handle occlusions and the range view representation of the scene changes drastically as agents nearby move. Our proposed framework overcomes these limitations by building a sparse 3D latent representation of the surroundings. We evaluate our method on SemanticKITTI, nuScenes, and PandaSet. Our experiments demonstrate the effectiveness of the proposed framework compared to the state-of-the-art. For more information, visit the project website: `https://waabi.ai/research/memoryseg`.

1. Introduction

Semantic segmentation of LiDAR point clouds is a key component for the safe deployment of self-driving vehicles (SDV). It enables SDVs to enhance their understanding of the surrounding environment by categorizing every 3D point into specific classes of interest, such as vehicles, pedestrians, traffic signs, buildings, roadways, etc. This rich and precise 3D representation of the environment can then be used for various applications such as generating online or offline semantic maps, building localization priors, or making the shapes of an object tracker more precise.

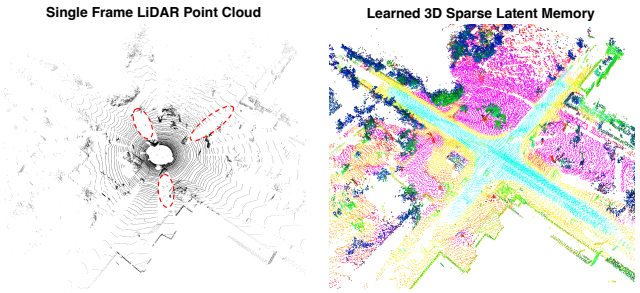

Figure 1. Objects could be partially occluded in single frame LiDAR point cloud. Our approach learns a 3D latent memory representation for better contextualizing the online observations. We apply PCA [14] to reduce the latent dimension to 3 and plot as RGB. (Best viewed in color and zoomed-in.)

LiDAR data is typically captured as a continuous stream of data, where every fraction of a second (typically 100ms) a new point cloud is available. Despite this fact, most LiDAR segmentation approaches process each frame independently [15, 32, 28, 22, 34, 5, 29] due to the computational and memory complexity associated with processing large amounts of 3D point cloud data. However, reasoning about a single frame suffers from the sparsity of the observations, particularly at range, and has difficulty handling occluded objects. Furthermore, the absence of motion information can make categorizing certain objects difficult.

Several approaches [20, 19] utilize a sliding window approach, where a small set of past frames is processed independently at every time step. The main shortcoming of this approach is its limited temporal context (typically under 1 second) due to resource constraints, as processing multiple LiDAR scans at once is expensive. TemporalLidarSeg [8] proposed to utilize a latent spatial memory to retain information about the past while avoiding redundant computation every time a new scan is available, in a range view (RV) representation. However, the RV of the scene changes drastically as the SDV or the other actors move due to changes in perspective. As a result, the memory can be dominated by nearby objects, which appear larger in the RV represen-

tation, making it challenging to be updated from frame to frame. Occlusion can be particularly challenging as the occluder now occupies the same spatial region that was previously describing the occluded object. This limits the usefulness of the memory. In contrast to RV, 3D is a metric space where the distances between points are preserved regardless of the viewpoint or relative distance to the SDV. Thus, representing the memory in 3D enables learning size priors for different classes. It allocates equal representation power to them regardless of the distance to the SDV, and enables easy understanding of motion. Moreover, previously observed regions that are currently occluded can be remembered in the memory as the occluder and occluded objects occupy different 3D regions, even though they share the same space in RV. Despite these advantages, 3D memory has been overlooked in LiDAR semantic segmentation.

In this paper we propose MEMORYSEG, a novel online LiDAR segmentation model that recurrently updates a sparse 3D latent memory as new observations are received, efficiently and effectively accumulating evidence from past observations. Fig. 1 illustrates the expressive power of such memory. In a single scan, objects are hard to identify due to sparsity (particularly at range) and lack of semantics, and occluded areas have no observations. In contrast, our latent memory is much denser, providing a rich context to separate different classes, especially in currently occluded regions.

To achieve an effective memory update that takes into account both the past and the present for accurate decoding, our method addresses several challenges. First, we align the previous memory with the current observation in the latest ego frame, to compensate for the SDV motion. Second, the sparsity level of the memory and the observation embeddings are different, making fusion non-trivial. The latent memory is denser, and thus some currently unobserved locations may be present in the memory. Complementarily, some regions in the current scan have not been observed before, such as new observations appearing as the SDV drives further. For this reason, we propose a mechanism to fill in missing regions in the current observations and add new observations to the memory during the update. Third, other objects are also moving (potentially in opposite direction), thus fusing the memory and observation embeddings requires a large receptive field. We address this via a carefully designed architecture that achieves a large receptive field yet preserves sparsity for efficiency. Aditionally, we introduce a novel point-level neighbourhood variation regularizer which penalizes significant differences in semantic predictions within local 3D neighborhoods.

Extensive experiments in SemanticKITTI [2], nuScenes [4] and PandaSet [27] demonstrate that MEMORYSEG outperforms current state-of-the-art semantic segmentation methods that rely purely on LiDAR on multiple benchmarks.

2. Related Work

In this section, we start by reviewing LiDAR semantic segmentation approaches in the literature and then focus on methods proposed for temporal LiDAR reasoning.

2.1. LiDAR Semantic Segmentation

LiDAR-based semantic segmentation methods can be classified into four categories based on the type of data being processed: point-based [16, 17, 13, 24], projection-based [28, 7, 32], voxel-based [5, 34, 12], or a combintation of the data representations [22, 29].

Point-based approaches [16, 17, 13, 24] operate directly on the point cloud, but most [17, 13, 24] rely on downsampling to accommodate limited computational resources. KPConv [24] introduces customized spatial kernel-based point convolution which yields the most favorable outcomes compared to other point-based methods. However, it require a significant amount of computational resources, which can limit its practicality in certain applications.

Projection-based methods [28, 7, 32] involve projecting a 3D point cloud onto a 2D image plane, which can be done using either in spherical Range-View [28, 7], Bird-Eye-View [32], or multi-view representations [11]. These approaches usually operate in real-time, benefiting from efficient 2D CNN inference on GPUs. However, they suffer from significant information loss due to projection, which can prevent them from being on par with state-of-the-art methods.

The emergence of 3D voxel-based approaches [5, 34, 12] can be attributed to recent advancements in sparse 3D convolutions [6, 21]. Typically, LiDAR point clouds are converted into Cartesian [5] or Cylindrical [34] voxels and then processed using sparse convolutions. They have demonstrated state-of-the-art performance, but reducing voxel resolution can result in significant loss of information.

Recently, researchers have explored the potential of leveraging multiple data representations [22, 30, 29, 31] to benefit from the information acquired by each representation. For instance, SPVNAS [22] incorporates both point-wise features and 3D-voxel features in their network while learning to segment. Similarly, RPVNet [29] utilizes a trilateral structure in their network, where each branch adopts a distinct data representation mentioned earlier.

Our method takes a similar approach to incorporate both point and voxel representations in the network. Voxel representations are utilized for contextual information learning, while point representations are employed to preserve fine-grained details, such as object boundaries and curvatures.

2.2. Temporal LiDAR Reasoning

Previous methods [20, 19, 26] have attempted to integrate a small number of past frames to facilitate temporal

reasoning. For instance, SpSequenceNet [20] introduces cross-frame global attention, which uses information from the previous frame to emphasize features in the current frame. Additionally, it utilizes cross-frame local interpolation to combine local features from both the previous and current frame. Further, MetaRangeSeg [26] includes information from past frames by having the residual depth map as an additional feature to the input. However, these methods are not well-suited for continuous application to a temporal sequence of LiDAR point clouds. Firstly, they are inefficient as they discard previous observations and require the model to start from scratch each time a new point cloud is received. Secondly, they only aggregate temporal information within a short horizon, failing to capture the potential of temporal reasoning over a longer duration.

Several works [8, 9] have attempted to incorporate temporal reasoning through a recurrent framework. Specifically, Duerr *et al*. [8] presents a recurrent segmentation architecture in RV, which uses previous observations as memory and aligns them temporally in the range image space. However, RV memory has limitations, such as prioritizing nearby objects, and suffering from contention of memory resources during occlusions (since information from behind the occluder and the occluder itself are now sharing the same memory entries). Further, StrObe [9] implemented a multi-scale 2D BEV memory for object detection enabling the network to take advantage of previous computations and process new LiDAR data incrementally. However, compared to object detection, semantic segmentation requires a more fine-grained and nuanced understanding of the scene and therefore a 3D memory is more suitable than 2D BEV.

Our approach is distinct from previous methods as we propose an online LiDAR semantic segmentation framework utilizing a sparse 3D memory. By representing the memory in 3D, we preserve the distance between points regardless of viewpoint or distance from the SDV, allowing for size priors of different classes to be learned. Additionally, the memory also retains previously observed regions even when currently occluded, as occluders and occluded objects occupy different 3D regions, despite being in the same space in RV. Furthermore, we preserve fine-grained height details that are lost in the BEV memory.

3. 3D Memory-based LiDAR Segmentation

In this section we introduce MEMORYSEG, an online semantic segmentation framework for streaming LiDAR point clouds that leverages a 3D latent memory to remember the past and better handle occlusions and sparse observations. In the remainder of this section, we first describe our model formulation, then present the network architecture and finally explain the learning process.

3.1. Model Formulation

Let $\mathcal{P} = \{\mathcal{P}_t\}_{t=1}^{L}$ be a sequence of LiDAR sweeps where $L \in \mathbb{N}^+$ is the sequence length and $t \in [1, L]$ is the time index. Each LiDAR sweep $\mathcal{P}_t = (\mathbf{G}_t, \mathbf{F}_t)$ is a 360° scan of the surrounding with N_t unordered points. $\mathbf{G}_t \in \mathbb{R}^{N_t \times 3}$ contains the Cartesian coordinates in the ego vehicle frame and $\mathbf{F}_t \in \mathbb{R}^{N_t}$ is the LiDAR intensity of the point. Let $\mathbf{T}_{t-1 \to t} \in SE(3)$ be the pose transformation from the vehicle frame at time $t-1$ to t.

To make informed semantic predictions, in this paper we maintain a latent (or hidden) memory in 3D. This memory is sparse in nature since the majority of the 3D space is unoccupied. To represent this sparsity, we parametrize the memory at time t using a sparse set of voxels containing the coordinates $H_{G,t} \in \mathbb{R}^{M_t \times 3}$ and their corresponding learned embeddings $H_{F,t} \in \mathbb{R}^{M_t \times d_m}$. M_t is the number of voxel entries in the latent memory at time t and d_m is the embedding dimension. Preserving the voxel coordinates is important to perform alignment as the reference changes when the SDV moves. We utilize a voxel-based sparse representation as it provides notable computational benefits with respect to dense tensors as well as sparse representations at the point level without sacrificing performance.

Inference follows a three-step process that is repeated every time a new LiDAR sweep is available: (i) the encoder takes in the most recent LiDAR point cloud at current time t and extracts point-level and voxel-level observation embeddings, (ii) the latent memory is updated taking into account the voxel-level embeddings from the new observations, and (iii) the semantic predictions are decoded by combining the point-level embeddings from the encoder and voxel-level embeddings from the updated memory. We refer the reader to Fig. 2 for an illustration of our method.

The memory update stage faces challenges due to the changing reference frame as the SDV moves, different sparsity levels of memory and current LiDAR sweep, as well as the motion of other actors. To address these challenges, a Feature Alignment Module (FAM) is introduced to align the previous memory state with the current observation embeddings. Subsequently, an Adaptive Padding Module (APM) is utilized to fill in missing observations in the current data and add new observations to the memory. Then, a Memory Refinement Module (MRM) is employed to update the latent memory using padded observations. Next, we explain each component in more detail.

Encoder: Following [22], our encoder is composed of a point-branch that computes point-level embeddings preserving the fine details and a voxel-branch that performs contextual reasoning through 3D sparse convolutional blocks [21]. The point-branch receives a 7-dimensional feature vector per point, with the xyz coordinates, intensity, and relative offsets to the nearest voxel center as features. It comprises two shared MLPs that output point embeddings,

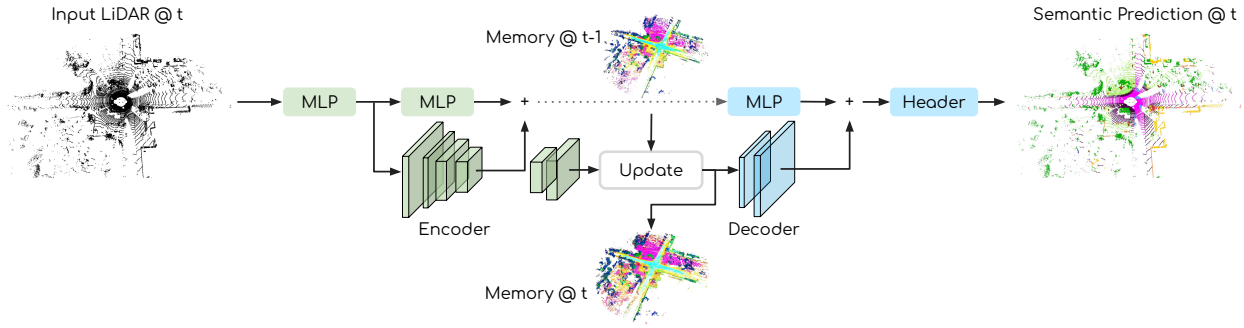

Figure 2. An overview of our model, MEMORYSEG. After the encoder processes the LiDAR point cloud at time t, the resulting feature map is utilized to update the latent memory (see Fig. 3 for more details about the memory update). Then, the decoder combines the refined memory with the point embeddings from the encoder to obtain semantic predictions.

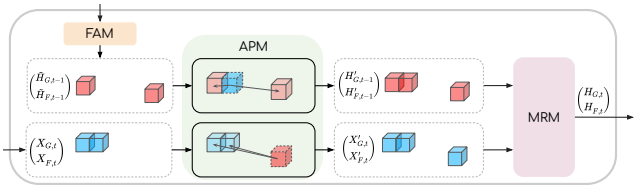

Figure 3. Overview of the latent memory update process. The latent memory embeddings ($H_{G,t-1}, H_{F,t-1}$) are transformed to the ego frame at t with the Feature Alignment Module (FAM). Next, the Adaptive Padding Module (APM) is utilized to learn the padding of both memory and observation embeddings. The Memory Refinement Module (MRM) updates the latent memory by incorporating the padded observation embeddings. The updated memory is then passed to the decoder for generating semantic predictions. (Best viewed in color and zoomed-in.)

as illustrated in Fig. 2. We average point embeddings belonging to the same voxel from the first shared MLP over voxels of size v_b to obtain voxel features. These features are then processed through four residual blocks with 3D sparse convolutions, each downsampling the feature map by a factor of 2. Two additional residual blocks with 3D sparse convolutions are applied to upsample the sparse feature maps. Unlike a full U-Net [18] that recovers features at the original resolution, we only upsample to $\frac{1}{4}$ of the original size for computational efficiency reasons, and use the coarser features to update the latent memory before decoding finer details to output our semantic predictions.

Feature Alignment: The reference frame changes as the SDV moves. We propose the Feature Alignment Module (FAM) to transform the latent memory from the ego frame at $t-1$ to t and align with the current observation embeddings. Specifically, we take the memory voxel coordinates $H_{G,t-1}$ and use the pose information $\mathbf{T}_{t-1\to t}$ to project from the ego frame at $t-1$ to t. We then re-voxelize using the projected coordinates with a voxel size of v_m. If multiple entries are inside the same memory voxel, we take the average of them to be the voxel feature. The resulting warped coordinates and embeddings of the memory in the ego frame t are denoted as $\tilde{H}_{G,t}$ and $\tilde{H}_{F,t}$, respectively.

Adaptive Padding: To handle the different sparsity of the latent memory and the voxel-level observation embeddings, we propose the Adaptive Padding Module (APM). We refer the readers to Fig. 3 for an illustration. First, we re-voxelize the encoder features with the same voxel size v_m where entries within the same voxel are averaged. We denote the resulting coordinates and embeddings as X_G and X_F. Note that we omit t in this section for brevity. Let $x_G \subseteq X_G$ and $x_F \subseteq X_F$ be the coordinates and embeddings of the new observations at time t that are not present in the memory. To obtain an initial guess of the memory embedding for a new entry, we utilize a weighted aggregation approach within its surrounding neighbourhood. This involves taking into account the coordinate offsets relative to the existing neighbouring voxels in the memory, which provides insight into their importance for aggregation, similar to Continuous Conv [25]. In addition to this, we incorporate the feature similarities and feature distances as additional cues for the aggregation process. Encoding feature similarities is particularly useful for assigning weights to the neighborhood. In a dynamic scene with moving actors, the closest voxel may not always be the most important voxel. By providing feature similarities, the network can make more informed decisions. The goal of such completion is to make a hypothesis of the embedding at the previously unobserved location using the available information. More precisely, we add those entries in the memory where their coordinates are $h'_G = x_G$ and the embedding of each voxel j are initialized as follows,

$$h'_{F,j} = \sum_{i \in \Omega_{\tilde{H}}(j)} w_{ji} \tilde{H}_{F,i}, \tag{1}$$

$$w_{ji} = \psi(\tilde{H}_{G,i} - x_{G,j}, \|\tilde{H}_{F,i} - x_{F,j}\|, \frac{\tilde{H}_{F,i} \cdot x_{F,j}}{\|\tilde{H}_{F,i}\| \|x_{F,j}\|}), \tag{2}$$

where i and j are voxel indices, $\Omega_{\tilde{H}}(j)$ is the k-nearest-neighbourhood of voxel j in $\tilde{H}_G$, and ψ is a shared MLP

followed by a softmax layer on the dimension of neighbourhood to ensure $\sum_i w_{ji} = 1$.

Second, we identify the regions in the memory that are unseen in the current observation and denote their coordinates and embeddings as $\tilde{h}_G \subseteq \tilde{H}_G$ and $\tilde{h}_F \subseteq \tilde{H}_F$. We add entries x'_G and x'_F to complete the current observation in a similar manner.

Memory Refinement: We design a sparse version of a ConvGRU [1] to update the latent memory $H'_{F,t-1}$ using the current padded observation embeddings $X'_{F,t}$ as follows:

$$\begin{aligned} r_t &= \text{sigmoid}[\Psi_r(X'_{F,t}, H'_{F,t-1})], \\ z_t &= \text{sigmoid}[\Psi_z(X'_{F,t}, H'_{F,t-1})], \\ \hat{H}_{F,t} &= \tanh[\Psi_u(X'_{F,t}, r_t \cdot H'_{F,t-1})], \\ H_{F,t} &= \hat{H}_{F,t} \cdot z_t + H'_{F,t-1} \cdot (1 - z_t), \end{aligned} \tag{3}$$

where Ψ_r, Ψ_z, Ψ_u are sparse 3D convolutional blocks with downsampling layers that aim to expand the receptive field and upsampling layers that restore the embeddings to their original size. r_t and z_t are learned signals to reset or update the memory, respectively. We refer the readers to the supplementary material about the detailed architecture of the sparse convolutional blocks.

Decoder: Our decoder consists of an MLP, two residual blocks with sparse 3D convolutions, and a linear semantic header. Specifically, we first take the corresponding memory embeddings at coordinates $\mathbf{G}_t$ and add with the point embeddings from the encoder. The resulting combined embeddings are then voxelized with voxel size of $\frac{v_b}{4}$ and further processed by two residual blocks that upsample the feature maps back to the original resolution. In parallel, an MLP takes the point embeddings before voxelization to retain the fine-grained details. Finally, the semantic header takes the combination of voxel and point embeddings to obtain per-point semantic predictions.

Memory Initialization: At the start of the sequence ($t = 0$), the first observation is used to initialize memory, with $H_{G,0} = X_{G,0}$ and $H_{F,0} = X_{F,0}$.

3.2. Learning

We learn our segmentation model by minimizing a linear combination of conventional segmentation loss functions [13, 34, 22] and a novel point-wise regularizer to better supervise the network training.

$$J = \beta_{wce} J_{wce} + \beta_{ls} J_{ls} + \beta_{reg} J_{reg}. \tag{4}$$

Here, J_{wce} denotes cross-entropy loss, weighted by the inverse frequency of classes, to address class imbalance in the dataset. The Lovasz Softmax Loss (J_{ls}) [3] is used to train the network, as it is a differentiable surrogate for the non-convex intersection over union (IoU) metric, which is a commonly used evaluation metric for semantic segmentation. Additionally, J_{reg} corresponds to our proposed point-wise regularizer. β_{reg}, β_{wce} and β_{ls} are hyperparameters.

Point-wise Smoothness: Our regularizer is designed to limit significant variations in semantic predictions within the 3D neighborhood of each point, except when these variations occur at the class boundary. Formally,

$$\begin{aligned} J_{reg} &= \frac{1}{N_t} \sum_{i=1}^{N_t} |\Delta(Y, i) - \Delta(\hat{Y}, i)|, \\ \Delta(Y, i) &= \Big| \frac{1}{|\Omega_{\mathcal{P}_t}(i)|} \sum_{j \in \Omega_{\mathcal{P}_t}(i)} |y_i - y_j| \Big|. \end{aligned} \tag{5}$$

Here, $\Delta(Y, i)$ represents the ground truth semantic variation around point i, while $\Delta(\hat{Y}, i)$ corresponds to the predicted semantic variation around point i. We use $\hat{Y} \in \mathbb{R}^{N_t \times C}$ to denote the predicted semantic distribution over C classes, and $Y \in \mathbb{R}^{N_t \times C}$ to denote the ground truth semantic one-hot label. The variable y_i represents the i-th element of Y. $\Omega_{\mathcal{P}_t}(i)$ denotes the neighborhood of point i in $\mathcal{P}_t$, and $|\Omega_{\mathcal{P}_t}(i)|$ represents the number of points in the neighborhood. The inspiration for this regularizer comes from a study by [11] that penalizes adjacent pixel prediction variations with the ground truth.

4. Experiments

In this section, we thoroughly analyze the performance of our method on three different datasets: SemanticKITTI [2], nuScenes [4], and PandaSet [27]. These datasets were collected using different LiDAR sensors and in diverse geographical regions. Our results demonstrate that MEMORY-SEG outperforms the current state of the art in all benchmarks. Additionally, we conduct ablation studies to understand the impact of our contributions. We demonstrate that incorporating a 3D sparse latent memory improves semantic predictions, with a very significant gain in long-range areas as these are sparser and more often partially obstructed than short-range regions.

Datasets: We benchmark our approach on three large-scale autonomous driving datasets: **SemanticKITTI** [2] consists of 22 sequences from the KITTI Odometry Dataset [10], which was collected in Germany using a Velodyne HDL-64E LiDAR. We use the standard split where sequences 0 to 10 are used for training (with sequence 8 for validation), and sequences 11 to 21 are held for testing. We follow the standard setting where semantic labels are mapped to 19 classes for the single-scan benchmark and 25 for the multi-scan benchmark. In the latter, all movable classes are further divided into moving and non-moving classes. Given that prior research on temporal aggregation for semantic segmentation is more prevalent in

Method	**mIoU**	Car	Bicycle	Motorcycle	Truck	Other Vehicle	Person	Bicyclist	Motorcyclist	Road	Parking	Sidewalk	Other Ground	Building	Fence	Vegetation	Trunk	Terrain	Pole	Traffic Sign	car (m)	bicyclist (m)	person (m)	motorcyclist (m)	other-vehicle (m)	truck (m)
TangentConv [23]	34.1	84.9	2.0	18.2	21.1	18.5	1.6	0.0	0.0	83.9	38.3	64.0	15.3	85.8	49.1	79.5	43.2	56.7	36.4	31.2	40.3	1.1	6.4	1.9	**30.1**	**42.2**
DarkNet53Seg [2]	41.6	84.1	30.4	32.9	20.2	20.7	7.5	0.0	0.0	91.6	64.9	75.3	27.5	85.2	56.5	78.4	50.7	64.8	38.1	53.3	61.5	14.1	15.2	0.2	28.9	37.8
KPConv [24]	51.2	93.7	44.9	47.2	42.5	38.6	21.6	0.0	0.0	86.5	58.4	70.5	26.7	90.8	64.5	84.6	70.3	66.0	57.0	53.9	69.4	67.4	67.5	47.2	4.7	5.8
Cylinder3D [34]	52.5	**94.6**	67.6	63.8	41.3	38.8	12.5	**1.7**	0.2	90.7	**65.0**	74.5	**32.3**	**92.6**	66.0	**85.8**	72.0	68.9	63.1	61.4	**74.9**	68.3	65.7	11.9	0.1	0.0
SpSequenceNet [20]	43.1	88.5	24.0	26.2	29.2	22.7	6.3	0.0	0.0	90.1	57.6	73.9	27.1	91.2	66.8	84.0	66.0	65.7	50.8	48.7	53.2	41.2	26.2	36.2	2.3	0.1
TemporalLidarSeg [8]	47.0	92.1	47.7	40.9	39.2	35.0	14.4	0.0	0.0	**91.8**	59.6	**75.8**	23.2	89.8	63.8	82.3	62.5	64.7	52.6	60.4	68.2	42.8	40.4	12.9	12.4	2.1
TemporalLatticeNet [19]	47.1	91.6	35.4	36.1	26.9	23.0	9.4	0.0	0.0	91.5	59.3	75.3	27.5	89.6	65.3	84.6	66.7	**70.4**	57.2	60.4	59.7	41.7	51.0	48.8	5.9	0.0
Meta-RangeSeg [26]	49.7	90.8	50.0	49.5	29.5	34.8	16.6	0.0	0.0	90.8	62.9	74.8	26.5	89.8	62.1	82.8	65.7	66.5	56.2	64.5	69.0	60.4	57.9	22.0	16.6	2.6
MEMORYSEG [ours]	**58.3**	94.0	**68.3**	**68.8**	**51.3**	**40.9**	**27.0**	0.3	**2.8**	89.9	64.3	74.8	29.2	92.2	**69.3**	84.8	**75.1**	70.1	**65.5**	**68.5**	71.7	**74.4**	**71.7**	**73.9**	15.1	13.6

Table 1. Comparison to the state-of-the-art methods on the test set of SemanticKITTI [2] multi-scan benchmark. (m) indicates moving. We include LiDAR-only published approaches at the time of submission. These approaches are categorized into two groups: the first group includes methods that aim for single-scan segmentation but use multiple aggregated frames as input; the second group consists of methods that introduce temporal aggregation modules explicitly tailored for handling multiple scans. Metrics are provided in [%].

the multi-scan benchmark, we consider it more appropriate for evaluating our proposed method. Single-scan benchmark results are also provided in the supplementary material. **nuScenes** [4] was collected in Boston and Singapore using a Velodyne HDL-32E LiDAR sensor, which captures sparser point clouds that pose extra challenges for semantic reasoning. The dataset contains a total of 1000 scenes, with 700 scenes used for training, 150 for validation, and the remaining 150 held out for testing. Each scene contains about 40 labeled frames recorded at 2 Hz. We follow the standard setting which maps 31 fine-grained classes to 16 classes for training and evaluation. **PandaSet** [27] provides short sequences of temporal data but with a large number of moving actors in Silicon Valley. This dataset was collected using a Hesai-Pandar64 LiDAR and consists of 103 sequences, each with a length of 8 seconds. Following [8], we grouped the original fine-grained semantic classes into 14 for training and testing, trying to match the classes in other popular segmentation benchmarks [2, 4]. We also used the same train/validation/test split as in the reference paper.

Implementation details: For all datasets, we first train the network without the memory update for single-scan segmentation for 50 epochs. Then, we freeze the encoder and train the memory update block and decoder in a recurrent fashion for an additional 20 epochs. During each training iteration, we use the first 10 consecutive sweeps as warmup memory, and then train the network with backpropagation through time (BPTT) on the next 3 frames. The memory voxel size v_m is 0.5 m, and the embedding dimension d_m is 128. We use the AdamW optimizer with a starting learning rate of 0.003 and a decay factor of 0.9. During training, we use data augmentation including global scaling sampled at random from [0.80, 1.20], translation sampled from [0, 0.2] m on all three axes, and global rotation around the Z axis with a random angle sampled from $[-\pi, \pi]$. We set the loss weights β_{wce} to 1, β_{ls} to 2, and β_{reg} to 500. We set the regularizer neighborhood $\Omega_{P_t}(i)$ to be the closest 32 points around point i, and use a padding neighbourhood $\Omega_{\tilde{H}}(j)$ to be the closest 5 voxels of voxel j. All experiments are conducted using 4 NVIDIA T4 GPUs with a batch size of 1 per GPU. During evaluation, we unroll each sequence from the first frame to the end and follow [34, 12] to apply test-time-augmentation, *i.e*. averaging prediction scores of 10 forward passes with augmented input from a single model.

To adapt to the uniqueness of each dataset, we consider the following per-dataset settings. To train on the SemanticKITTI dataset, we set the voxel size v_b to 0.05 m. Due to the limited quantities of movable actors in this dataset, following [33] we generate a library of instances from the training sequences and randomly inject 5 objects over the ground classes at each frame during training. To improve the ability to separate classes into moving and non-moving actors, we add another linear header to classify points as moving or static and fuse the results from the semantic header. More implementation details are provided in the supplementary material. For nuScenes, the voxel size is set to 0.1 m, and 0.125 m for PandaSet.

Metrics: We follow existing works [34, 5, 29] and use the intersection over union (IoU) averaged over all classes as the main metric (mIoU). Class-wise IoUs are also reported.

Comparison against state-of-the-art: Our results are compared with other state-of-the-art approaches on the test set of three datasets in Tab. 1 to 3. We include approaches that use LiDAR only for fair comparison. Notably, our proposed method outperforms all prior works on all three datasets, demonstrating its generalizability across various LiDAR sensors and different geographical regions.

Tab. 1 presents a comparison of our approach on the SemanticKITTI multi-scan benchmark, which is designed to evaluate methods that focus on temporal aggregation for semantic segmentation. The other methods in the table are divided into two groups. The first group consists of baselines that are designed for single-scan segmentation but aggregate multiple consecutive frames as input at the cost of higher memory consumption and longer runtime. In contrast, the second group proposes temporal aggregation modules specifically to handle multiple scans. It is worth noting that our proposed method performs significantly better than all of these approaches. MEMORYSEG demonstrates exceptional performance in segmenting dynamic actors, such as *moving bicyclist*, *motorcyclist*, and *person*. These substantial improvements can be attributed to the proposed re-

Method	mIoU	FW-mIoU	Barrier	Bicycle	Bus	Car	Construction	Motorcycle	Pedestrain	Traffic Cone	Trailer	Truck	Drivable	Other Flat	Sidewalk	Terrain	Manmade	Vegetation
PolarNet [32]	69.4	87.4	72.2	16.8	77.0	86.5	51.1	69.7	64.8	54.1	69.7	63.5	96.6	67.1	77.7	72.1	87.1	84.5
Cylinder3D [34]	77.2	89.9	82.8	29.8	84.3	89.4	63.0	**79.3**	77.2	73.4	84.6	69.1	97.7	**70.2**	80.3	75.5	90.4	87.6
SPVCNN [22]	77.4	89.7	80.0	30.0	**91.9**	90.8	64.7	79.0	75.6	70.9	81.0	74.6	97.4	69.2	80.0	76.1	89.3	87.1
(AF)2-S3Net [5]	78.3	88.5	78.9	**52.2**	89.9	84.2	**77.4**	74.3	77.3	72.0	83.9	73.8	97.1	66.5	77.5	74.0	87.7	86.8
MEMORYSEG [ours]	**80.6**	**91.4**	**84.9**	40.2	91.2	**92.4**	71.2	73.5	**85.9**	**77.8**	**88.0**	**76.4**	**97.9**	69.0	**81.2**	**77.6**	**92.6**	**89.7**

Table 2. Comparison to the state-of-the-art methods on the test set of nuScenes [4] LiDAR semantic segmentation benchmark. We include LiDAR-only published approaches at the time of submission. Metrics are provided in [%]

Method	mIoU	Car	Bicycle	Motorcycle	Truck	Other Vehicle	Person	Road	Road Barriers	Sidewalk	Building	Vegetation	Terrain	Background	Traffic Sign
SqueezeSegv3 [28]	55.7	92.8	24.1	18.0	36.5	54.3	63.0	91.1	11.9	71.3	86.2	85.0	61.3	63.2	20.6
SalsaNext [7]	57.8	92.1	40.7	31.7	28.7	56.2	69.0	90.0	22.6	67.1	85.6	83.4	58.5	63.3	20.6
TemporalLidarSeg [8]	60.0	93.7	33.6	38.0	37.1	59.9	72.0	91.1	14.6	70.6	88.2	88.4	63.8	68.4	20.7
SPVCNN [22]	64.7	95.8	38.1	46.3	44.0	74.1	78.6	91.2	**28.3**	70.3	87.2	87.5	61.5	67.5	35.6
MEMORYSEG [ours]	**70.3**	**97.2**	**60.2**	**58.4**	**62.9**	**74.3**	**82.6**	**92.1**	27.7	**74.1**	**89.4**	**90.7**	**64.9**	**72.8**	**36.4**

Table 3. Comparison to the state-of-the-art methods on the test set of PandaSet [27]. Metrics are provided in [%].

current framework, which implicitly encode the motion of moving actors in the 3D latent memory. Furthermore, we also demonstrate the effectiveness of our approach on the single-scan benchmark of the same dataset. We refer the readers to the supplementary material for the detailed results. This is a more competitive benchmark focusing on single-scan semantic segmentation where previous research has focused on proposing various architectures [5, 29, 34] or knowledge distillation techniques [12]. Our results show that MEMORYSEG can outperform these methods, which are highly optimized for this benchmark.

Results on nuScenes [4] benchmark are presented in Tab. 2. MEMORYSEG outperforms the previous state-of-the-art. Our method is particularly effective in smaller classes such as *barriers*, *pedestrians*, and *traffic cones*, which present challenges for semantic segmentation networks due to the sparser point clouds available in this dataset. However, our approach overcomes this limitation by using a 3D latent memory to improve semantic reasoning of the sparse points. Additionally, we observe significant gains in the background classes, such as *sidewalk*, *terrain*, and *manmade*, which often rely on understanding of the surrounding to be segmented correctly. Our method improves the contextual reasoning by accumulating past observations using a latent memory representation.

Finally, we compare our results with the state-of-the-art methods on PandaSet [27] and report the test set IoUs in Tab. 3. Our proposed method also achieves state-of-the-art performance in this dataset. Notably, our approach outperforms others by a large margin in almost all classes. We highlight that MEMORYSEG consistently outperforms TemporalLidarSeg [8] across multiple benchmarks, a method that also proposes a latent memory but in RV instead of 3D. These findings confirm our hypothesis that the 3D latent memory approach is more effective than using RV in accurately segmenting objects in 3D point clouds.

Importance of the memory: Fig. 4 highlights the significance of incorporating memory, particularly in longer range regions where point clouds are much sparser. To assess the effectiveness of the learned latent memory, we compare against our encoder and decoder without the memory update module as the single frame baseline (SFB). Our proposed network consistently outperforms the SFB in all regions, with the most significant improvement observed in the long-range region of nuScenes, where points are sparser and more difficult to contextualize. Thanks to the 3D sparse latent memory that accumulates semantic embeddings of past frames, our proposed approach performs much better than the SFB, particularly in those challenging regions. Fig. 5 provides similar insights. In the top-left of the figure, the SFB fails to segment a partially occluded vehicle located behind a fence. In contrast, our proposed approach initially segments the vehicle as *other vehicle* class, but as the ego vehicle advances, the 3D latent memory accumulates and refines through new observations, enabling the network to correctly identify the vehicle as a *car* after two sweeps. This illustration further confirms that the proposed 3D latent memory provides better contextualization of sparse objects that are partially occluded. Additional qualitative results are provided in the supplementary materials. We also quantitatively demonstrate the importance of memory in Tab. 4. M1 builds on SFB and incorporates FAM and MRM. Note that both FAM and MRM are essential for updating the latent memory continuously. FAM aligns the sparse 3D latent memory in the traveling ego frame, while MRM helps in learning to reset and update the memory recurrently. APM is replaced with zero-padding in M1 to highlight the significance of memory. M1 shows a 2.3% improvement in the mIoU metric compared to SFB, attributed to the ability of the sparse 3D latent memory to aggregate information in the

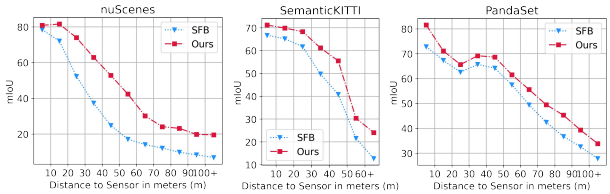

Figure 4. Comparison of MEMORYSEG with single frame baseline (SFB) on the validation set with different distance-range.

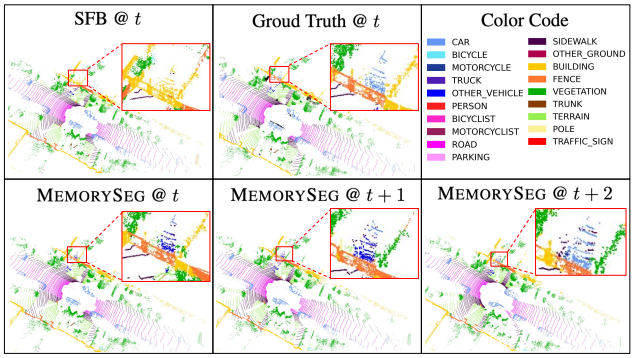

Figure 5. Predictions of MEMORYSEG over time are illustrated in the bottom row. We include the prediction from the single frame baseline (SFB) and ground truth with color code at the top.

temporal dimension as the ego moves through the scene.

Influence of adaptive padding: In this section, we compare our proposed APM with zero padding and Continuous Conv [25]. The results are presented in Tab. 4. M2 is an extension of M1 that introduces adaptive padding on both observation and memory embeddings when they are aligned. However, the padded entries are only processed using the relative coordinates in the neighborhood, similar to Continuous Conv [25] with the attention mechanism. A slight improvement of 0.2% is observed when new observations in the latent memory are initialized with a learned guess instead of all zeros. To further improve the performance of the network, M4 exploits feature similarities and distances in APM. When assigning an initial embedding to a new entry, both the relative position and the similarity to neighbors are considered. This is particularly useful for moving actors in the scene, where the network should learn to start with a guess with a similar embedding in the neighborhood, rather than the closest entry. By comparing M0 with M3 further confirms the usefulness of the proposed APM.

Influence of memory refinement: In M3 of Tab. 4, we replace our proposed MRM with a vanilla sparse ConvGRU [1], which has a limited receptive field due to the convolution kernel size, making it challenging for fast-moving actors. However, MRM overcomes this limitation by downsampling features to increase the receptive field. This improves the learning of meaningful reset and update gates for refining the latent memory.

	FAM	APM		MRM		mIoU
		CC	Ours	SCG	Ours	
SFB						67.2
M0	✓	✓		✓		68.9
M1	✓				✓	69.5
M2	✓	✓			✓	69.7
M3	✓		✓	✓		69.7
M4	✓		✓		✓	**70.8**

Table 4. Ablation on the proposed components in the network. Metrics are mIoU on the validation set of SemanticKITTI [2] provided in [%]. **FAM**: Feature Alignment Module, **APM**: Adaptive Padding Module, **CC**: padding using Continuous Conv [25], **SCG**: Sparse ConvGRU [1]. **MRM**: Memory Refinement Module.

Regularizer	SemanticKITTI	nuScenes	PandaSet
	66.4	72.9	64.7
✓	**67.2**	**76.7**	**66.6**

Table 5. Ablation on the model performance with and without the proposed regularizer. Metrics are mIoU on the validation set provided in [%].

Influence of the regularizer: To demonstrate the effectiveness of our regularizer to train semantic segmentation networks on point clouds with different characteristics, we ablate this loss on the three datasets and summarize the results in Tab. 5. The results consistently show that incorporating the regularizer leads to performance improvements. The regularizer provides additional supervision on variations and boundaries, which is particularly beneficial for nuScenes. We note once again that this dataset has sparser point clouds than the other two.

Parameter count and runtime comparison: MEMORYSEG incurs only a relatively small computational overhead of 23% more parameters and 20% runtime than SFB. On the other hand, when naively concatenating the past five frames as the input to the model, despite having the same number of parameters as SFB, it takes 1.58 times longer (32% slower than our method) while only exploiting a limited 0.5-second window from the past.

5. Conclusion

In this paper, we have proposed a novel online LiDAR segmentation model named MEMORYSEG, which utilizes a sparse 3D latent memory to recurrently accumulate learned semantic embeddings from past observations. We also presented a novel variation regularizer to supervise the learning of 3D semantic segmentation on point clouds. Our results demonstrate that our approach achieve state-of-the-art performance on three large-scale datasets. This improved performance can be attributed to the ability of the latent memory to better contextualize objects in the scene. Looking ahead, our future work will focus on integrating instance segmentation and tracking to develop an end-to-end memory-augmented panoptic segmentation framework.

References

[1] Nicolas Ballas, Li Yao, Chris Pal, and Aaron C. Courville. Delving deeper into convolutional networks for learning video representations. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[2] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jürgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *2019 IEEE/CVF International Conference on Computer Vision*, pages 9296–9306. IEEE, 2019.

[3] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4413–4421, 2018.

[4] 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 *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11621–11631, 2020.

[5] Ran Cheng, Ryan Razani, Ehsan Taghavi, Enxu Li, and Bingbing Liu. af2-s3net: Attentive feature fusion with adaptive feature selection for sparse semantic segmentation network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12547–12556, 2021.

[6] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3075–3084, 2019.

[7] Tiago Cortinhal, George Tzelepis, and Eren Erdal Aksoy. Salsanext: Fast semantic segmentation of lidar point clouds for autonomous driving. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 655–661, 2020.

[8] Fabian Duerr, Mario Pfaller, Hendrik Weigel, and Jürgen Beyerer. Lidar-based recurrent 3d semantic segmentation with temporal memory alignment. In *2020 International Conference on 3D Vision (3DV)*, pages 781–790. IEEE, 2020.

[9] Davi Frossard, Shun Da Suo, Sergio Casas, James Tu, and Raquel Urtasun. Strobe: Streaming object detection from lidar packets. In Jens Kober, Fabio Ramos, and Claire Tomlin, editors, *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 1174–1183. PMLR, 16–18 Nov 2021.

[10] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[11] Martin Gerdzhev, Ryan Razani, Ehsan Taghavi, and Liu Bingbing. Tornado-net: multiview total variation semantic segmentation with diamond inception module. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9543–9549, 2021.

[12] Yuenan Hou, Xinge Zhu, Yuexin Ma, Chen Change Loy, and Yikang Li. Point-to-voxel knowledge distillation for lidar semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8479–8488, 2022.

[13] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11108–11117, 2020.

[14] Ian Jolliffe. *Principal Component Analysis*. John Wiley & Sons, Ltd, 2005.

[15] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. Rangenet++: Fast and accurate lidar semantic segmentation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[16] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proc. of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[17] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[18] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[19] Peer Schütt, Radu Alexandru Rosu, and Sven Behnke. Abstract flow for temporal semantic segmentation on the permutohedral lattice. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2022.

[20] Hanyu Shi, Guosheng Lin, Hao Wang, Tzu-Yi Hung, and Zhenhua Wang. Spsequencenet: Semantic segmentation network on 4d point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4574–4583, 2020.

[21] Haotian Tang, Zhijian Liu, Xiuyu Li, Yujun Lin, and Song Han. Torchsparse: Efficient point cloud inference engine. *Proceedings of Machine Learning and Systems*, 4:302–315, 2022.

[22] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d architectures with sparse point-voxel convolution. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII*, pages 685–702. Springer, 2020.

[23] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3d. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3887–3896, 2018.

[24] 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 *Proc. of the IEEE International Conference on Computer Vision*, pages 6411–6420, 2019.

[25] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2589–2597, 2018.

[26] Song Wang, Jianke Zhu, and Ruixiang Zhang. Meta-rangeseg: Lidar sequence semantic segmentation using multiple feature aggregation. *IEEE Robotics and Automation Letters*, 7(4):9739–9746, 2022.

[27] Pengchuan Xiao, Zhenlei Shao, Steven Hao, Zishuo Zhang, Xiaolin Chai, Judy Jiao, Zesong Li, Jian Wu, Kai Sun, Kun Jiang, et al. Pandaset: Advanced sensor suite dataset for autonomous driving. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 3095–3101. IEEE, 2021.

[28] 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 *European Conference on Computer Vision*, pages 1–19. Springer, 2020.

[29] Jianyun Xu, Ruixiang Zhang, Jian Dou, Yushi Zhu, Jie Sun, and Shiliang Pu. Rpvnet: A deep and efficient range-point-voxel fusion network for lidar point cloud segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16024–16033, 2021.

[30] Xu Yan, Jiantao Gao, Chaoda Zheng, Chao Zheng, Ruimao Zhang, Shuguang Cui, and Zhen Li. 2dpass: 2d priors assisted semantic segmentation on lidar point clouds. In *European Conference on Computer Vision*, pages 677–695. Springer, 2022.

[31] Maosheng Ye, Shuangjie Xu, Tongyi Cao, and Qifeng Chen. Drinet: A dual-representation iterative learning network for point cloud segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7447–7456, 2021.

[32] 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 *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9601–9610, 2020.

[33] Zixiang Zhou, Yang Zhang, and Hassan Foroosh. Panoptic-polarnet: Proposal-free lidar point cloud panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13194–13203, 2021.

[34] Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Yuexin Ma, Wei Li, Hongsheng Li, and Dahua Lin. Cylindrical and asymmetrical 3d convolution networks for lidar segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9939–9948, 2021.