2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Exploiting Local Features and Range Images for Small Data Real-Time Point Cloud Semantic Segmentation

Daniel Fusaro, Simone Mosco, Emanuele Menegatti and Alberto Pretto

***Abstract*— Semantic segmentation of point clouds is an essential task for understanding the environment in autonomous driving and robotics. Recent range-based works achieve real-time efficiency, while point- and voxel-based methods produce better results but are affected by high computational complexity. Moreover, highly complex deep learning models are often not suited to efficiently learn from small datasets. Their generalization capabilities can easily be driven by the abundance of data rather than the architecture design. In this paper, we harness the information from the three-dimensional representation to proficiently capture local features, while introducing the range image representation to incorporate additional information and facilitate fast computation. A GPU-based KDTree allows for rapid building, querying, and enhancing projection with straightforward operations. Extensive experiments on SemanticKITTI and nuScenes datasets demonstrate the benefits of our modification in a "small data" setup, in which only one sequence of the dataset is used to train the models, but also in the conventional setup, where all sequences except one are used for training. We show that a reduced version of our model not only demonstrates strong competitiveness against full-scale state-of-the-art models but also operates in real-time, making it a viable choice for real-world case applications. The code of our method is available at `https://github.com/Bender97/WaffleAndRange`.**

I. INTRODUCTION

Point cloud semantic segmentation is a challenging task that has gained considerable interest in recent years, mainly due to its application in relevant fields such as autonomous driving, robotics, and environmental perception [1]. In particular, accurate semantic segmentation of the surrounding environment is essential for decision-making even in real-time applications such as autonomous vehicle navigation and intelligent robot navigation. However, achieving a balance between accuracy and computational efficiency remains a considerable challenge. An even greater challenge lies in achieving accurate semantic segmentation when the available dataset is small. In such cases, the learning model must demonstrate exceptional generalization capabilities to accurately segment unseen data. Generally, highly complex state-of-the-art models tend to perform comparably well or even worse than low-complexity neural networks [2].

Recent approaches have made significant advancements in the field by leveraging various LiDAR point cloud representations, including point-based [3]–[5], voxel-based [6], pseudo-image [7]–[9], and hybrid representation [10]–[12]. Notably, hybrid representations generally outperform other classes of methods by taking advantage of the strengths of each representation. However, while the richness of information benefits performance, it introduces drawbacks in terms of processing time, making them unsuitable for real-world scenarios.

{fusarodani, moscosimon, emg, alberto.pretto}@dei.unipd.it
Department of Information Engineering, University of Padova, Italy.

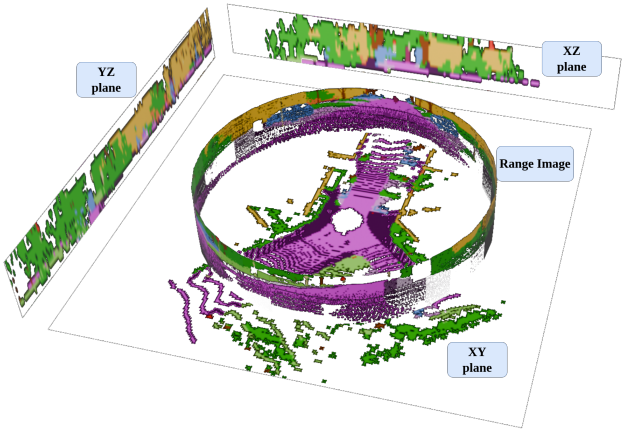

Fig. 1. The four 2D projections utilized by our system for semantically segmenting the 3D point cloud are as follows: XY, XZ, YZ, and range image projection.

Range image representations allow the use of established 2D convolutional neural networks (CNNs), which require less processing time but often produce less accurate results due to information loss caused by the projection of 3D points onto a 2D plane. In contrast, point and voxel-based methods benefit from the original 3D representation, enabling the capture of local features and point distribution. However, these methods require significantly longer computation times. In this work, we rely on both the aforementioned representations by combining the strengths of two recent approaches, WaffleIron [13] and RangeFormer [9], to achieve real-time efficiency and enhance generalization capabilities. Our network comprises three main components: a point cloud embedding module, a backbone composed of point cloud processing layers, and a segmentation head for generating final predictions. Specifically, we explore the utility of the point embedding module in WaffleIron, which encompasses richer information, operates at a three-dimensional level, and leverages local features, geometry, and shape characteristics. Simultaneously, we harness the range image in the backbone as a valuable representation, facilitating the use of dense 2D convolution and ensuring faster computational time. In addition, we prioritize the optimization of inference time. We introduce a GPU-based KDTree for building and querying inside the point cloud embedding module, offering

time savings compared to the CPU version. Furthermore, we enhance the flattening operations proposed in WaffleIron through straightforward element-wise multiplication and scattering operations. This ensures faster computation and lower memory consumption because we do not use large and very sparse matrices for this operation.

The key contributions of this work are as follows:

- We introduce a novel deep learning architecture integrating two state-of-the-art models, WaffleIron and Rangeformer, designed to improve the quality of semantic segmentation for automotive point clouds in scenarios with limited data availability.
- We demonstrate through extensive experiments that the extraction of local features within the embedding module, coupled with the integration of range image data, significantly enhances performance in scenarios characterized by limited data availability (hereinafter referred to as "small data" setup). Furthermore, our approach yields meaningful improvements even in contexts where abundant data is available, thus exhibiting its versatility across different data regimes.
- We drastically reduce the full-model system runtime to just 180 ms per point cloud without any inference optimizer, thanks to code optimization and a GPU-based KDTree. We show that a reduced version of our model shows strong competitiveness with full-scale state-of-the-art models but also operates in real-time, with a runtime of 80 ms.
- We conduct an in-depth performance evaluation of our and other recent methods on two autonomous driving benchmarks: SemanticKITTI [14] and nuScenes [15].
- An open-source implementation of the proposed method is released for public usage.

II. RELATED WORKS

Point cloud semantic segmentation methods can be divided into four paradigms: *point-based*, *projection-based*, *discretization-based*, and *hybrid methods*.

A. Point-based Methods

Point-based methods directly work on the raw unordered point cloud. PointNet [16] is the pioneer in this field, followed by its improved version PointNet++ [17]. They approximate a symmetric function using multi-layer perceptrons (MLPs) to obtain permutation invariance of the input points and learn per-point features. KPConv [3], DGCNN [18] and PointConv [19] apply convolution-like operations to exploit local geometric features. In RandLA-Net [4] the point cloud is first subsampled while an attention mechanism retains relevant information. PointASNL [5] introduces an adaptive sampling module to deal with point clouds with noise. WaffleIron [13] proposes a novel 3D backbone, relying on MLPs over 3D points and dense 2D convolution on projection planes. Recent works [20]–[22] rely on the self-attention mechanism introduced in transformers [23] and directly work on the 3D points. However, point-based methods are rarely used to process LiDAR point clouds since they are generally time-consuming.

B. Projection-based Methods

These methods usually rely on a 2D representation of the point cloud by projecting it onto a surface. SqueezeSeg [24] proposes an encoder-decoder network based on SqueezeNet [25] and conditional random fields to refine the predictions. SqueezeSegv2 [26] and SqueezeSegv3 [27] introduce respectively a context aggregation module and a spatially-adaptive convolution to improve the network. In [28] a top-view image is extracted and fed to a fully connected network, while in [29] bird's eye view projections are exploited. On the opposite side, SalsaNet [30] and its evolution SalsaNext [31] claim that the projection type does not bring any advantage to the segmentation of their work. RangeNet++ [7] integrates DarkNet as the backbone to leverage range images and proposes an efficient k-NN post-processing technique. KPRNet [32] exploits KPConv as segmentation head while Lite-HDSeg [33] introduces three different modules and a boundary loss to improve the results. FIDNet [34] and CENet [35] switch encoders to ResNet [36] and substitute the decoders with simple interpolation. Recent works as RangeViT [8] exploit vision transformers [37] pre-trained on image data, while RangeFormer [9] achieves state-of-the-art performance introducing a pyramidal structure based on [38]. Projection-based approaches usually leverage accelerated computations by operating almost entirely within the 2D image space, thereby attaining real-time efficiency.

C. Discretization-based Methods

These methods first discretize the 3D point cloud into voxel representation and then leverage 3D convolution operators. In [39] the input is voxelized but the label predictions are refined at a per-point level. MinkowskiNet [40] uses sparse convolution [41] to reduce the computational cost while $(AF)^2$-S3Net [42] aggregates multi-scale features with an attention mechanism. Cylinder3D [6] uses a cylindrical grid to partition the space and designs an asymmetrical network to predict labels. Recent studies [43] consider the varying sparsity in LiDAR point clouds, introducing a self-attention mechanism with a radial window. Discretization-based methods leverage the geometric properties of the 3D space, yet they incur higher computational costs when dealing with outdoor LiDAR point clouds.

D. Hybrid Methods

Recent trends combine multiple representations (points, projection images, and voxels) or RGB images from camera sensors to boost performance. In [44]–[46] fine-grained features at point level are integrated with high-level voxel representations. RPVNet [10] introduces a range-point-voxel fusion network that leverages information from the three distinct representations. PVKD [11] improves the performance by introducing a point-voxel knowledge distillation module while 2DPASS [12] uses RGB images to enrich semantic and structural information during the training process.

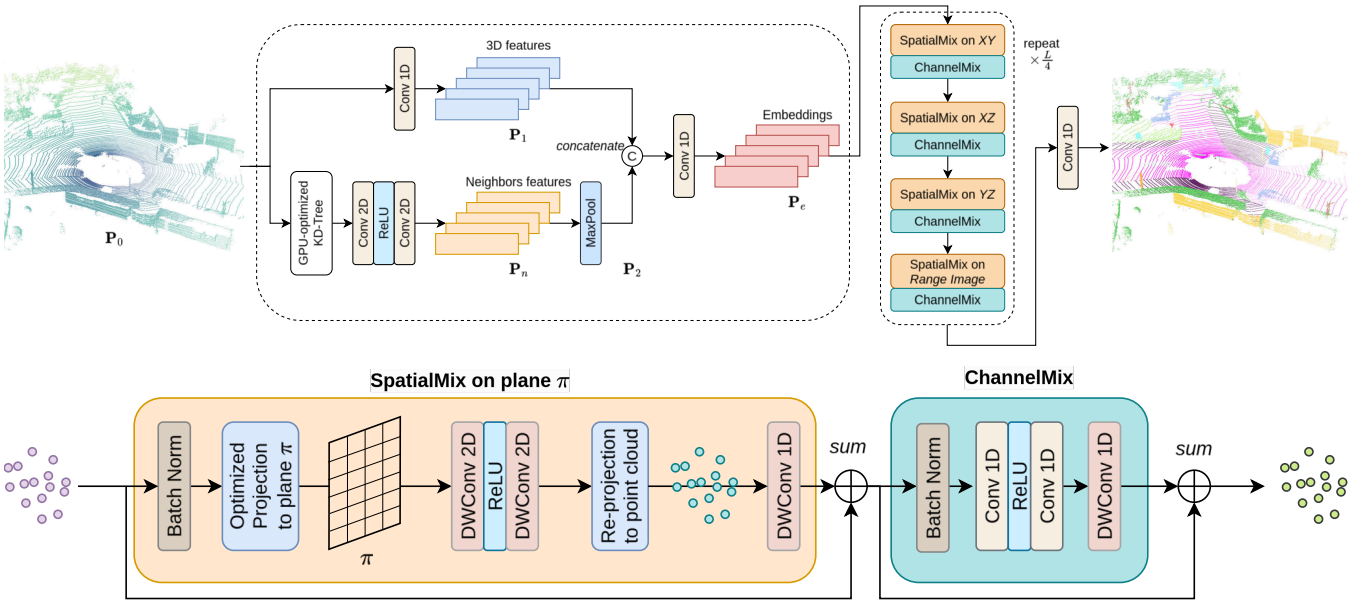

Fig. 2. (Above) An architectural overview of the proposed method featuring Point Cloud Embedding for point features computation, Point Cloud processing layers as the backbone with integrated Spatial and channel mix modules, and a Segmentation Head for generating final predictions. (Below) A detailed representation of the backbone, showcasing the spatial mix and channel mix modules. The spatial mix includes batch normalization, projection onto a 2D plane, 2D depth-wise convolution, re-projection onto 3D points, 1D depth-wise convolution, and a residual connection. Meanwhile, the channel mix employs batch normalization, 1D convolution, 1D depth-wise convolution, and a residual connection.

Recent works as FRNet [47] combine range image pixels and frustum LiDAR points to boost point-level predictions while UniSeg [48] enriches voxel, range, and point-based representations with color and texture information from RGB images. Hybrid approaches leverage the strengths of each representation and may achieve a balanced trade-off between accuracy and efficiency.

III. METHOD

Fig. 2 provides a detailed overview of our method. Based on the recent state-of-the-art works WaffleIron [13] and RangeFormer [9], it leverages the best of the two paradigms to produce high-quality point cloud segmentation and reduce the runtime of the system. It consists of three main stages: the point cloud embedding (pre-processing), the feature sharing stage, and the final segmentation head.

A. Point Cloud Embedding

Both WaffleIron and RangeFormer consist of these three stages, at least from a high-level point of view, implementing them in different flavors. Based on theoretical arguments and empirical evidence (see Section V), we found that WaffleIron's embedding gives a much more informative representation than RangeFormer's. The latter pre-processing stage consists of a module of 3 multi-layer perceptrons (MLPs), called range embedding module (REM), that directly operates on the range image built from the input point cloud. It maps the range image from $\mathbb{R}^{B\times 6\times H\times W}$, where B is the batch size, H and W are the range image's height and width respectively, to a higher dimensional space, $\mathbb{R}^{B\times C\times H\times W}$. The three steps are 6 to 64, 64 to 128, and 128 again to 128. It also consists of batch norm and Gaussian error linear unit (GELU) activation in each of the three MLPs.

WaffleIron's pre-processing does not operate on the range image but directly on the input point cloud. Firstly, the input point cloud is voxelized and cropped to the sensor field of view, reducing sensor outliers and ignoring points that are too far away. At this point, we obtain a point cloud $\mathbf{P_0} \in \mathbb{R}^{N\times 5}$, where N is the number of points in the point cloud and 5 is the dimensionality of each point's feature, namely x, y, z, LiDAR intensity value, and distance from the sensor (calculated as the Euclidean distance). $\mathbf{P_0}$ is processed by a simple 1D Convolutional layer that maps the five features to 128, resulting in $\mathbf{P_1}$. Then, a K-Dimensional Tree (KDTree) is built over the point cloud $\mathbf{P_0}$. We use this data structure to efficiently retrieve the K-Nearest Neighbors (kNN) to each point and build a local feature. Similar to PointNet [16], the local feature extraction captures information such as shape characteristics, local geometry, and point distribution patterns. This hierarchical processing enables the network to learn representations that are robust to variations in scale, orientation, permutation, and density.

The tensor point cloud of neighbors $\mathbf{P_n} \in \mathbb{R}^{K\times N\times 5}$ is obtained by stacking, for each point in $\mathbf{P_0}$, the K neighbors' features. As depicted in Fig. 2, $\mathbf{P_n}$ is processed by a series of batch norm, 2D convolutions, and rectified linear unit (ReLU) activation functions. Following this processing, a max pooling operation is applied over the K neighbors to select only the maximum response. This operation preserves the most salient features while suppressing noise, enhancing the representation's robustness to noisy input, and contribut-

ing to the permutation and rotation invariance of the network. Let's denote the output of this processing as the point cloud $\mathbf{P_2} \in \mathbb{R}^{N\times 128}$, which has the same number of points and feature size as $\mathbf{P_1}$, thanks to the pooling operation.

Finally, the two point clouds $\mathbf{P_1}$ and $\mathbf{P_2}$ are concatenated and remapped to a feature size of 128 using a 1D convolution. Let this final point cloud be denoted as $\mathbf{P_e} \in \mathbb{R}^{N\times 128}$.

B. Leveraging 2D and 1D Convolutions

The preprocessed point cloud $\mathbf{P_e}$ is then iteratively parsed by a sequence of spatial mix and channel mix operations. These manipulate 2D projections of the 3D point cloud and simply employ 1D or 2D convolutions. 3D convolutions are intentionally avoided to reduce the complexity of the model and thus mitigate overfitting drawbacks. Simpler models are generally more stable and tend to better generalize to unseen datasets (see Section V for an empirical evaluation).

In this setup, the 3D data processing stage is reduced to the simpler, faster, and more efficient processing of 2D images. 2D convolutions excel in image semantic segmentation by capturing spatial information and relationships within an image. Unlike fully connected layers, which treat pixels independently, 2D convolutions consider local patterns and structures through kernel convolution. This facilitates learning features such as edges, textures, and shapes vital for object segmentation. Moreover, shared weights of convolutional kernels ensure parameter efficiency and translation invariance, making them ideal for capturing hierarchical representations crucial in semantic segmentation tasks.

1D convolutions are not commonly used for image semantic segmentation tasks due to their inability to capture the intricate spatial relationships present in images. Images inherently possess complex two-dimensional structures with spatial arrangements of pixels, and 1D convolutions are more suited for tasks involving sequential data. Through kernel convolution across a signal, they can extract meaningful features and relationships, aiding tasks such as denoising.

The 2D projections (see Fig. 1) of the point cloud comprise orthogonal projections onto the planes XY, XZ, and YZ, as implemented in the original work WaffleIron, augmented with the 2D range image as seen in RangeFormer. The former consists of simple 2D grids where 3D points are projected, and cells are vectors in $\mathbb{R}^{128}$ obtained by averaging the features of points within the same cell. Traditionally, the latter, as seen in RangeFormer, represents the point cloud by encoding the distance of each point from the sensor frame. Typically, only the closest point in each cell is considered for range image construction, resulting in information loss. In our work, however, we adopt the concept of feature averaging, similar to the three orthogonal projections, by considering all points belonging to the same cell of the range image.

A sequence of L independent but identical modules parses these 2D projections. Each layer processes a specific 2D projection using the following procedure:

- Project the points onto the specified 2D plane (XY, XZ, YZ, or Range Image).
- Perform the spatial mix operation over the 2D projection.
- Conduct point cloud feature inflation, where 3D points within the same 2D cell inherit the features of that cell.
- Execute the channel mix operation over the inflated 3D point cloud points.

The spatial mix and channel mix modules are illustrated in Fig. 2. The spatial mix module comprises two 2D depthwise convolutions separated by a ReLU activation function, followed by a 1D grouped convolution. Similarly, the channel mix module mirrors the spatial mix module, with the 2D convolutions replaced by a 1D convolution. Both modules begin with an initial batch norm of the input and end with a skip connection consisting in the summing of the output of the module and the input.

C. Segmentation Head

To fully exploit the local features collected in the preprocessing step, we merge the parsed point cloud with the kNN embedding by simply summing them up. This skip connection offers evident benefits, such as enhancing the contextualization of point segmentation and facilitating gradient updates to the embedding module during the backpropagation phase.

The final labels are inferred by applying a 1D convolution over the merged point cloud followed by a Softmax layer.

IV. EXPERIMENTS

A. Implementation Details

We implemented our method in Python, C++, and CUDA. All the inference experiments were performed on a consumer laptop PC equipped with an AMD Ryzen 7 5800H CPU (3.2 GHz), 16GB of RAM, and Linux OS (the internal GPU has not been used). All the deep learning training were done on a Desktop PC with an Intel Core i9-10920X CPU (3.50 GHz), 32 GB of RAM, an Nvidia Titan RTX GPU, and a Linux OS. All the results reported refer to the performance obtained at the 45th epoch of the training process.

B. Inference Time Reduction: KDTree on GPU

As outlined in Section III, the pre-processing module necessitates constructing a KDtree over the input point cloud. Initially implemented by the authors of WaffleIron using the KDTree class from the SciPy library [49], this approach proved fast on CPU but fell short of real-time requirements. Consequently, a GPU-boosted implementation emerged as a desirable alternative. By adapting cudaKDTree[1], an open-source library written in C++ and CUDA for constructing and querying KD-trees, we transitioned the computation to the GPU. Leveraging GPU optimization, the total time required for both building and querying the tree reduced drastically from approximately 160 ms to a mere 15 ms.

[1]https://github.com/ingowald/cudaKDTree

TABLE I

SEMANTIC SEGMENTATION PERFORMANCE ON SEMANTICKITTI VALIDATION SET (SEQUENCE 8) OF METHODS TRAINED ONLY ON SEQUENCE 04. THE ✗ SYMBOL INDICATES CLASSES NOT PRESENT IN THE TRAINING SET.

method	**mIoU %**	*car*	*bicycle*	*motorcycle*	*truck*	*other-vehicle*	*person*	*bicyclist*	*motorcyclist*	*road*	*parking*	*sidewalk*	*other-ground*	*building*	*fence*	*vegetation*	*trunk*	*terrain*	*pole*	*traffic-sign*
RangeFormer [9]	15.6	47.9	✗	✗	✗	0.2	0.1	✗	✗	62.5	0.1	27.9	1.3	26.6	1.8	57.4	2.6	41.1	15.6	11.8
FRNet [47]	22.0	71.3	✗	✗	✗	0.6	0.1	✗	✗	65.2	0.2	47.4	1.9	52.3	2.0	76.8	2.2	59.5	14.5	24.3
Cylinder3D [6]	23.7	68.7	✗	✗	✗	0.8	1.7	✗	✗	64.0	0.3	44.3	1.1	57.5	8.5	78.8	16.4	57.9	24.6	25.3
WaffleIron [13]	24.6	86.8	✗	✗	✗	0.1	1.5	✗	✗	60.8	0.9	43.6	0.5	55.1	4.8	82.0	18.7	67.6	25.5	19.2
ours	**24.9**	86.4	✗	✗	✗	0.2	1.2	✗	✗	61.9	0.5	47.3	0.5	55.9	4.7	80.4	18.5	66.9	24.9	23.4

TABLE II

SEMANTIC SEGMENTATION PERFORMANCE ON SEMANTICKITTI VALIDATION SET (SEQUENCE 8) OF METHODS TRAINED ONLY ON SEQUENCE 04 (SECOND COLUMN) OR FULL TRAINING SET (EXCEPT SEQUENCE 8, THIRD COLUMN).

method	**Small data mIoU %**	**Full data mIoU %**
Rangeformer [9]	15.9	67.9
FRNet [47]	22.5	68.7
Cylinder3D [6]	23.2	64.3
WaffleIron [13]	24.6	68.0
ours	**24.9**	**69.0**

C. Inference Time Reduction: Flattening Operation

The original implementation of WaffleIron, in the spatial mix module, employs a batch matrix multiplication technique to average the features of points located within the same cell. This process involves constructing a large matrix in $\mathbb{R}^{B,HW,N}$, where B represents the batch size, HW denotes the flattened size of the grid under consideration, and N is the total number of points of the point cloud. Subsequently, this matrix is multiplied by a weight matrix of size $\mathbb{R}^{N,B}$, with each element representing the inverse density of the corresponding cell that the point occupies. While technically accurate, this approach imposes significant demands on both time and memory resources since the initial matrix is highly sparse. The multiplication process, even when performed on a GPU, incurs substantial time overhead. The required time of each of these multiplications considering all the $L = 48$ layers is 130 ms.

By rearranging the sequence of these operations, we managed to reduce the total time required for all the layers to 90 ms, with a net advantage of 40 ms. Instead of generating a large sparse matrix, we first perform an element-wise multiplication between the features of the point cloud and the weight matrix. Subsequently, we aggregate the weighted features using a scattering operation. This operation can be efficiently executed using the *scatter_add_* function provided by the torch library [50]. By supplying the indices of the cell to which each weighted point feature should be added, we rapidly construct the final matrix. In the end, we were able to reduce the total runtime from 365 ms to 180 ms. As described in Section V, we obtained competitive results using only L=12 layers, but running *real-time* in 80 ms.

D. Datasets

We evaluate our method on two extensive autonomous driving datasets: SemanticKITTI [14] and nuScenes [15].

SemanticKITTI comprises 22 sequences, with each point cloud segmented into 19 semantic classes. We adopted the standard split where the initial 11 sequences formed the training set, with the exception of the 8th sequence used for validation. This sequence is the most complete and variegated. The remaining 11 sequences constituted the test set.

In nuScenes, each point is labeled with one of the 16 considered semantic classes. The dataset comprises 1000 scenes, but, following the official division, we used 700 scenes for training, 150 for validation, and 150 for testing.

E. Evaluation Metrics

As commonly done in Semantic Segmentation tasks, we report the Intersection-over-Union (IoU) for class i and the average score (mIoU) over all classes, where $\text{IoU}_i = \frac{\text{TP}_i}{\text{TP}_i+\text{FP}_i+\text{FN}_i}$. TP_i , FP_i and FN_i are the true positives, false positives, and false negatives.

F. Performance on Autonomous Driving Datasets

The performance evaluation is conducted in two different setups. The first setup aims to evaluate the generalization capabilities of the models in scenarios where data is scarce. Referred to as the "small data" setup, this approach recognizes that while a vast amount of available data is typically beneficial for deep learning models, it may not be sufficient to truly evaluate their efficiency. Even a well-trained model on a large dataset may not have an optimal architecture, yet still deliver excellent results on the test set. To address this, we initially train using only a single, concise set of point cloud data. Specifically, these trainings are conducted solely on SemanticKITTI sequence 04, with performance validation on sequence 08. Sequence 04 is the smallest of the training dataset, consisting of only 271 point clouds. In contrast, sequence 08 comprises 4071 point clouds, making the models' learning very challenging. Data augmentations applied in this setup include random X-Y flipping and/or rotation, along with random scaling with a maximum scale factor of 0.1.

TABLE III

SEMANTIC SEGMENTATION PERFORMANCE ON NUSCENES VALIDATION SET. *: REGARDING 2DPASS, WE REPORT THE RESULTS OF THE BASELINE OF [12] TRAINED WITH LIDAR DATA BUT NO IMAGES, I.E., IN THE SAME SETTING AS THE OTHER METHODS IN THIS TABLE.

method	**mIoU %**	*barrier*	*bicycle*	*bus*	*car*	*const. veh.*	*motorcycle*	*pedestrian*	*traffic cone*	*trailer*	*truck*	*driv. surf.*	*other flat*	*sidewalk*	*terrain*	*manmade*	*vegetation*
RangeNet++ [7]	65.5	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
PolarNet [29]	71.0	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
SalsaNext [31]	72.2	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
Cylinder3D [6]	76.1	76.4	40.3	91.2	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
2DPASS* [12]	76.2	75.3	43.5	95.3	91.2	54.5	78.9	72.8	62.1	70.0	83.2	96.3	73.2	74.2	74.9	88.1	85.9
WaffleIron [13]	77.6	78.7	51.3	93.6	88.2	47.2	86.5	81.7	68.9	69.3	83.1	96.9	74.3	75.6	74.2	87.2	85.2
ours	**77.6**	78.5	49.6	91.8	87.6	52.7	86.7	82.2	70.1	67.2	79.7	97.0	74.7	76.8	74.9	87.5	85.0

TABLE IV

SMALL DATA SETUP. RESULTS ON METHODS TRAINED ONLY ON SEQUENCE 04 OF SEMANTICKITTI (271 POINT CLOUDS) AND TESTED ON SEQUENCE 08 (4071 POINT CLOUDS).

method	range	L	drop	skip	aug	**mIoU %**
original	✗	48	-	-	✗	24.4
ours	✓	48	✗	✗	✗	**24.9**
ours	✓	48	✓	✓	✗	22.3
ours	✓	48	✗	✓	✗	23.8
original	✗	48	-	-	✓	29.3
ours	✓	48	✗	✗	✓	32.7
ours	✓	48	✗	✓	✓	**32.7**
original	✗	12	-	-	✓	28.4
ours	✓	12	✓	✓	✓	20.6
ours	✓	12	✗	✗	✓	28.5
ours	✓	12	✗	✓	✓	**29.8**

TABLE V

RESULTS ON METHODS TRAINED ON FULL TRAINING SET OF SEMANTICKITTI AND TESTED ON SEQUENCE 08 (VALIDATION SET). NETWORK CONFIGURATION WITH ONLY 12 LAYERS AND ACTIVE DATASET AUGMENTATION.

method	range	L	drop	skip	aug	**mIoU %**
original	✗	12	-	-	✓	62.6
ours	✓	12	✗	✓	✓	**65.8**

We retrained the models to validate their performance on the "full data" setup, where the full datasets are utilized. The trainings were conducted on both SemanticKITTI and nuScenes datasets, with the latter exclusively utilized in the full data setup. Our models were trained with $L = 48$ layers. We employed 256-dimensional point features and a grid resolution of 40 cm for SemanticKITTI, while for nuScenes, we used $F = 384$ and a grid resolution of 60 cm. Additionally, exclusively for SemanticKITTI, following the data augmentation applied in the small data setup, we incorporated the InstanceCutMix and PolarMix techniques utilized by the authors of WaffleIron.

The results of the small data setup are reported in Table I. It shows a clear improvement of our method with respect to all the other state-of-the-art methods, including the original architecture of WaffleIron. This is due to the much more informative view given from the range image, which helps the model to better share the features during the Waffle processing. It is able to improve the results of 0.3% of mIoU with respect to the original WaffleIron model.

Our method is also the best performing in the validation sequence of the full data setup (see Table II), in which it obtains a mIoU of 69.0% and improves the results of the original WaffleIron of 1.0%. This confirms the generalization capabilities that the model obtains in the small data setup. In Section V, we conduct a more in-depth analysis of the various components that led us to the architecture development.

In Table III, we present the results obtained on the validation set of nuScenes. The table clearly indicates the superiority of both the original WaffleIron and our method, with our approach achieving slightly better performance on average. However, the observed benefit is not as pronounced as in SemanticKITTI. This discrepancy may be attributed to the fact that the range image projection is applied to point clouds obtained from a LiDAR with 32 channels, compared to the 64 channels used in SemanticKITTI. Consequently, the range image provides less informative data, making object distinctions less clear. It's worth noting that our method demonstrates improved segmentation of fundamental semantic objects such as motorcycles, pedestrians, and traffic cones. Furthermore, we achieve top results in classes related to construction vehicles, drivable surfaces, sidewalks, terrain, and manmade structures.

V. ABLATION STUDY

In this section, an ablation study conducted on the SemanticKITTI dataset is carried out and described.

Table IV reports the results of the experiments on the small data setup. On the top block, we only applied data augmentations such as random X-Y flip and/or rotation and random scaling, with a maximum scaling factor of 0.1. The columns drop, skip and aug refer, respectively, to three techniques that we tested. The first is a random dropout of the neighbor points used in the embedding module. The second is the application of a skip connection between the neighbor features computed from the embedding module and the processed points computed by the backbone. In other words, we just sum up the neighbors embedding and the final point cloud processing just before the final classification of the

points. The third is the application of the data augmentation techniques InstanceCutMix and PolarMix that WaffleIron authors used. It's important to note that these introduce also instances that are not present in the sequence 04 that we use for training, taking these instances from the other sequences. These tests are a good indicator of the performance of the methods on the full training set.

From the results shown in Table IV, it can be observed that the dropout technique does not significantly improve the robustness of the model. However, skip connections exhibit interesting behavior: while they marginally decrease performance without augmentation, they lead to performance enhancement when InstanceCutMix and PolarMix augmentations are applied, yielding a mIoU of 32.7%.

Furthermore, we analyzed the performance of the method when using a reduced number of layers ($L = 12$). The results, summarized in Table V, mirror the observations made with the full model configuration: dropout tends to decrease performance, whereas skip connections tend to enhance it. Motivated by these findings, we conducted a comprehensive study with $L = 12$ layers on the entire training set, validating the results on sequence 08. Remarkably, our method achieved a mIoU of 65.8%, surpassing the baseline method WaffleIron's mIoU (62.6%). This performance is comparable to methods employing $L = 48$ layers. Additionally, our method meets real-time requirements with a runtime of less than 100 ms, averaging at 80 ms.

VI. CONCLUSIONS

We addressed the challenge of achieving real-time, accurate semantic segmentation of point clouds in cases where the training dataset is small. Building upon two existing state-of-the-art models, our architecture proficiently leverages local feature extraction during point cloud embedding and hybrid projection of both the 3D point cloud and the range image, thus increasing the accuracy and efficiency. Through extensive analysis on benchmarks like SemanticKITTI and nuScenes, our experiments show that incorporating local features into the embedding module and integrating range image data greatly boosts performance in situations with scarce data. Additionally, our method also improves results in data-rich contexts, exhibiting its versatility across various data scenarios. Through code optimization we significantly decreased the system runtime, providing a robust solution for real-world semantic segmentation tasks for autonomous robotics.

REFERENCES

[1] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3d point clouds: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 12, pp. 4338–4364, 2020.

[2] L. Brigato and L. Iocchi, "A close look at deep learning with small data," in *2020 25th International Conference on Pattern Recognition (ICPR)*. Los Alamitos, CA, USA: IEEE Computer Society, jan 2021, pp. 2490–2497. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/ICPR48806.2021.9412492

[3] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "Kpconv: Flexible and deformable convolution for point clouds," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6411–6420.

[4] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, "Randla-net: Efficient semantic segmentation of large-scale point clouds," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 108–11 117.

[5] X. Yan, C. Zheng, Z. Li, S. Wang, and S. Cui, "Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 5589–5598.

[6] X. Zhu, H. Zhou, T. Wang, F. Hong, Y. Ma, W. Li, H. Li, and D. Lin, "Cylindrical and asymmetrical 3d convolution networks for lidar segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 9939–9948.

[7] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "Rangenet++: Fast and accurate lidar semantic segmentation," in *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2019, pp. 4213–4220.

[8] A. Ando, S. Gidaris, A. Bursuc, G. Puy, A. Boulch, and R. Marlet, "Rangevit: Towards vision transformers for 3d semantic segmentation in autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 5240–5250.

[9] L. Kong, Y. Liu, R. Chen, Y. Ma, X. Zhu, Y. Li, Y. Hou, Y. Qiao, and Z. Liu, "Rethinking range view representation for lidar segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 228–240.

[10] J. Xu, R. Zhang, J. Dou, Y. Zhu, J. Sun, and S. 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*, 2021, pp. 16 024–16 033.

[11] Y. Hou, X. Zhu, Y. Ma, C. C. Loy, and Y. Li, "Point-to-voxel knowledge distillation for lidar semantic segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 8479–8488.

[12] X. Yan, J. Gao, C. Zheng, C. Zheng, R. Zhang, S. Cui, and Z. Li, "2dpass: 2d priors assisted semantic segmentation on lidar point clouds," in *European Conference on Computer Vision*. Springer, 2022, pp. 677–695.

[13] G. Puy, A. Boulch, and R. Marlet, "Using a waffle iron for automotive point cloud semantic segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 3379–3389.

[14] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences," in *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019.

[15] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *CVPR*, 2020.

[16] C. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 77–85, 2017.

[17] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

[18] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *ACM Transactions on Graphics (tog)*, vol. 38, no. 5, pp. 1–12, 2019.

[19] W. Wu, Z. Qi, and L. Fuxin, "Pointconv: Deep convolutional networks on 3d point clouds," in *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, 2019, pp. 9621–9630.

[20] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, "Point transformer," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 16 259–16 268.

[21] X. Wu, Y. Lao, L. Jiang, X. Liu, and H. Zhao, "Point transformer v2: Grouped vector attention and partition-based pooling," *Advances in Neural Information Processing Systems*, vol. 35, pp. 33 330–33 342, 2022.

[22] X. Wu, L. Jiang, P.-S. Wang, Z. Liu, X. Liu, Y. Qiao, W. Ouyang, T. He, and H. Zhao, "Point transformer v3: Simpler, faster, stronger," *arXiv preprint arXiv:2312.10035*, 2023.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[24] B. Wu, A. Wan, X. Yue, and K. Keutzer, "Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 1887–1893.

[25] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[26] B. Wu, X. Zhou, S. Zhao, X. Yue, and K. Keutzer, "Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 4376–4382.

[27] C. Xu, B. Wu, Z. Wang, W. Zhan, P. Vajda, K. Keutzer, and M. Tomizuka, "Squeezesegv3: Spatially-adaptive convolution for efficient point-cloud segmentation," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII 16*. Springer, 2020, pp. 1–19.

[28] L. Caltagirone, S. Scheidegger, L. Svensson, and M. Wahde, "Fast lidar-based road detection using fully convolutional neural networks," in *2017 ieee intelligent vehicles symposium (iv)*. IEEE, 2017, pp. 1019–1024.

[29] Y. Zhang, Z. Zhou, P. David, X. Yue, Z. Xi, B. Gong, and H. Foroosh, "Polarnet: An improved grid representation for online lidar point clouds semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9601–9610.

[30] E. E. Aksoy, S. Baci, and S. Cavdar, "Salsanet: Fast road and vehicle segmentation in lidar point clouds for autonomous driving," in *2020 IEEE intelligent vehicles symposium (IV)*. IEEE, 2020, pp. 926–932.

[31] T. Cortinhal, G. Tzelepis, and E. Erdal Aksoy, "Salsanext: Fast, uncertainty-aware semantic segmentation of lidar point clouds," in *Advances in Visual Computing: 15th International Symposium, ISVC 2020, San Diego, CA, USA, October 5–7, 2020, Proceedings, Part II 15*. Springer, 2020, pp. 207–222.

[32] D. Kochanov, F. K. Nejadasl, and O. Booij, "Kprnet: Improving projection-based lidar semantic segmentation," *arXiv preprint arXiv:2007.12668*, 2020.

[33] R. Razani, R. Cheng, E. Taghavi, and L. Bingbing, "Lite-hdseg: Lidar semantic segmentation using lite harmonic dense convolutions," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 9550–9556.

[34] Y. Zhao, L. Bai, and X. Huang, "Fidnet: Lidar point cloud semantic segmentation with fully interpolation decoding," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 4453–4458.

[35] H.-X. Cheng, X.-F. Han, and G.-Q. Xiao, "Cenet: Toward concise and efficient lidar semantic segmentation for autonomous driving," in *2022 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2022, pp. 01–06.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[37] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[38] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao, "Pyramid vision transformer: A versatile backbone for dense prediction without convolutions," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 568–578.

[39] L. Tchapmi, C. Choy, I. Armeni, J. Gwak, and S. Savarese, "Segcloud: Semantic segmentation of 3d point clouds," in *2017 international conference on 3D vision (3DV)*. IEEE, 2017, pp. 537–547.

[40] C. Choy, J. Gwak, and S. Savarese, "4d spatio-temporal convnets: Minkowski convolutional neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 3075–3084.

[41] B. Graham, M. Engelcke, and L. Van Der Maaten, "3d semantic segmentation with submanifold sparse convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9224–9232.

[42] R. Cheng, R. Razani, E. Taghavi, E. Li, and B. 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*, 2021, pp. 12 547–12 556.

[43] X. Lai, Y. Chen, F. Lu, J. Liu, and J. Jia, "Spherical transformer for lidar-based 3d recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 17 545–17 555.

[44] H. Tang, Z. Liu, S. Zhao, Y. Lin, J. Lin, H. Wang, and S. Han, "Searching efficient 3d architectures with sparse point-voxel convolution," in *European conference on computer vision*. Springer, 2020, pp. 685–702.

[45] F. Zhang, J. Fang, B. Wah, and P. Torr, "Deep fusionnet for point cloud semantic segmentation," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIV 16*. Springer, 2020, pp. 644–663.

[46] J. Park, C. Kim, S. Kim, and K. Jo, "Pcscnet: Fast 3d semantic segmentation of lidar point cloud for autonomous car using point convolution and sparse convolution network," *Expert Systems with Applications*, vol. 212, p. 118815, 2023.

[47] X. Xu, L. Kong, H. Shuai, and Q. Liu, "Frnet: Frustum-range networks for scalable lidar segmentation," *arXiv preprint arXiv:2312.04484*, 2023.

[48] Y. Liu, R. Chen, X. Li, L. Kong, Y. Yang, Z. Xia, Y. Bai, X. Zhu, Y. Ma, Y. Li, *et al.*, "Uniseg: A unified multi-modal lidar segmentation network and the openpcseg codebase," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 21 662–21 673.

[49] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[50] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.