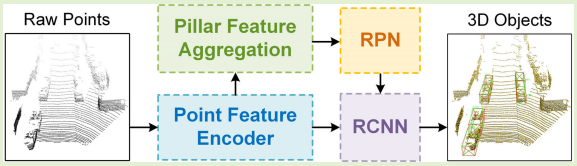
HybridPillars: Hybrid Point-Pillar Network for Real-Time Two-Stage 3-D Object Detection

Zhicong Huang, Yuxiao Huang, Zhijie Zheng, Haifeng Hu, *Member, IEEE*, and Dihu Chen, *Member, IEEE*

***Abstract*—LiDAR-based 3-D object detection is an important perceptual task in various fields such as intelligent transportation, autonomous driving, and robotics. Existing two-stage point-voxel methods contribute to the boost of accuracy on 3-D object detection by utilizing precise point-wise features to refine 3-D proposals. Although obtaining promising results, these methods are not suitable for real-time applications. First, the inference speed of existing point-voxel hybrid frameworks is slow because the acquisition of point features from voxel features consumes a lot of time. Second, existing point-voxel methods rely on 3-D convolution for voxel feature learning, which increases the difficulty of deployment on embedded computing platforms. To address these issues, we propose a real-time two-stage detection network, named HybridPillars. We first propose a novel hybrid framework by integrating a point feature encoder into a point-pillar pipeline efficiently. By combining point-based and pillar-based networks, our method can discard 3-D convolution to reduce computational complexity. Furthermore, we propose a novel pillar feature aggregation network to efficiently extract bird's eye view (BEV) features from pointwise features, thereby significantly enhancing the performance of our network. Extensive experiments demonstrate that our proposed HybridPillars not only boosts the inference speed, but also achieves competitive detection performance compared to other methods. The code will be available at https://github.com/huangzhicong3/HybridPillars.**



***Index Terms*—3-D object detection, LiDAR point clouds, real time, two-stage.**

I. Introduction

LIDAR-BASED 3-D object detection is an important perceptual task in various fields such as intelligent transportation, autonomous driving, and robotics [1], [2], [3], [4]. However, this task is still challenging due to the sparsity and irregularity of point clouds generated by LiDAR sensors. Existing studies can be divided into two categories: single-stage methods and two-stage methods. Two-stage 3-D detection frameworks [5], [6], [7], [8], [9], [10], [11], [12], [13] generate a set of candidate proposals in the first stage and employ region-based convolutional neural networks (RCNNs) as the second-stage network to refine 3-D proposals. PV-RCNN [5] first proposes a two-stage pipeline that adopts a voxel-based backbone for proposal generation and refines the 3-D boxes with pointwise features. Although promising results have been reported, its slow inference speed and high computational cost make it difficult to apply to real-time application. Deng et al. [14] find that almost half of the inference time is consumed by the voxel set abstraction (SA) layer in PV-RCNN [5]. As shown in Fig. 1(a), the voxel SA layer is to learn the pointwise features from the voxel features. Accurate point features are important for two-stage point-voxel frameworks since they provide fine-grained 3-D information for refinement. Therefore, to improve the computational efficiency of two-stage detection, the first issue is to optimize the network architecture and accelerate the acquisition of point features.

Apart from the slow inference speed, 3-D convolution is another factor that limits the application of existing two-stage detection frameworks. Most high-performance two-stage frameworks rely on 3-D convolutional networks for voxel feature extraction. SECOND [15] implements the sparse 3-D convolutional layers to accelerate the voxel-based feature learning networks. The sparse 3-D convolution has been widely applied in existing two-stage frameworks that utilize the 3-D voxel representation for feature learning. However, when developing real-time applications on edge computing devices, the 3-D convolutional layers will greatly increase the difficulty of model deployment [16]. To meet this demand, PointPillars [17] proposes to use 2-D representation with a pillar-based pipeline for 3-D object detection task. As shown in Fig. 1(b), PointPillars converts the raw point clouds into 2-D bird's eye view (BEV) feature maps during the voxelization to

Received 12 May 2024; revised 8 September 2024 and 21 September 2024; accepted 22 September 2024. Date of publication 2 October 2024; date of current version 14 November 2024. This work was supported by the Science and Technology Program of Guangdong Province under Grant 2022B0701180001. The associate editor coordinating the review of this article and approving it for publication was Prof. Ling Pei. *(Corresponding author: Dihu Chen.)*

Zhicong Huang, Yuxiao Huang, Zhijie Zheng, and Haifeng Hu are with the School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou 510006, China.

Dihu Chen is with the School of Integrated Circuits, Sun Yat-sen University, Shenzhen 518000, China (e-mail: stscdh@mail.sysu.edu.cn).

Digital Object Identifier 10.1109/JSEN.2024.3468646

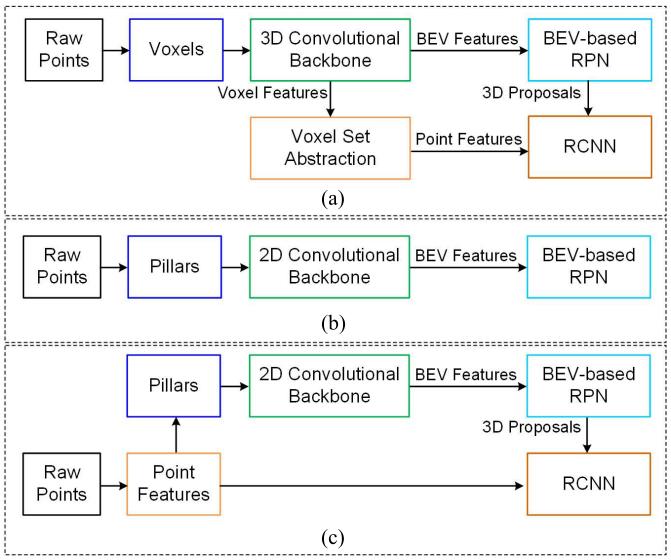


Fig. 1. Workflows of the (a) PV-RCNN, (b) PointPillars, and (c) our HybridPillars.

discard 3-D convolution. There are also some methods [16], [18], [19], [20], [21] following this pillar-based pipeline and achieving satisfying performance on 3-D object detection.

Motivated by the pillar-based methods, we propose a two-stage hybrid framework, HybridPillars, which utilizes both the point-based and pillar-based pipelines in a single network. Our workflow is illustrated in Fig. 1(c). First, the raw point clouds pass through a point feature encoder for preliminary feature learning in our method. In contrast to previous pillar-based methods, our pipeline can encode precise 3-D information into the feature vectors of key points. Afterward, the pointwise features are projected into the BEV plane for 3-D proposal generation with a pillar-based network. In the second stage, we take both the raw points and the learned point features from the encoder to refine proposals and generate final detection results.

Compared with previous methods, the main advantage of our model is that no additional cost is needed for point feature extraction. Moreover, our model is deployment-friendly for edge computing platform since our hybrid point-pillar network only requires vanilla 2-D convolutional layers to detect 3-D objects. To improve the performance of our network, we further propose an effective pillar feature aggregation network, which is designed to learn the BEV features from the pointwise features. It consists of a nonempty pillar generator, an attentive pillar feature extraction (PFE) module, and an accelerated BEV backbone. The proposed nonempty pillar generator can propagate the gradient from the pillar-based network to the point-based network, enabling end-to-end training of our network. The attentive PFE module can enhance the grouped pillar features and improve the detection performance. Finally, the accelerated BEV backbone utilizes the sparsity of input point features to speed up the inference. Extensive experiments on the KITTI detection benchmark [22] and the ONCE dataset [23] show that HybridPillars achieves competitive detection performance and fast inference speed compared with the related methods.

We summarize our contributions into three facets.

1) We present a novel 3-D object detection framework, named HybridPillars, which utilizes a hybrid network to obtain accurate point features for proposal generation and refinement.
2) We propose a pillar feature aggregation network that consists of a nonempty pillar generator, an attentive PFE module, and an accelerated BEV backbone. This network can not only learn BEV features from pointwise features efficiently, but also improve the accuracy of our hybrid point-pillar network.
3) Extensive experiments on the KITTI detection benchmark and ONCE dataset demonstrate that HybridPillars achieves competitive performance on 3-D object detection while running fast. Moreover, our two-stage framework is deployment-friendly because the hybrid point-pillar network does not involve 3-D convolution for feature learning.

II. Related Work

A. Grid-Based Methods

LiDAR sensors generate point clouds that offer precise 3-D spatial information. To efficiently process the irregular and unstructured point clouds, grid-based methods convert the point clouds to regular grids before applying the convolutional neural networks.

1) Voxel Representations: VoxelNet [24] builds a deep neural network with 3-D convolution to process the voxels. SECOND [15] employs sparse 3-D convolution to reduce redundant computations on empty voxels, thus improving the efficiency of the voxel-based detection network. Voxel R-CNN [14] aggregates the fine-grained features of 3-D proposals and refines them in a voxel-based refinement network. BtcDet [25] predicts the occupied voxels with a shape-learning network, which helps to locate the objects under occlusion. VoxelNeXt [26] employs a fully sparse convolutional network architecture, effectively decoupling the process of sparse-to-dense conversion. VoxelMamba [27] proposes a group-free strategy to serialize the 3-D voxels for voxel-based 3-D detection.

2) Pillar Representations: PointPillars [17] expands the voxel in the Z-axis during the voxelization to generate the pseudo-image feature maps on BEV. Thus, only 2-D convolution is needed in [17] to generate 3-D bounding boxes. This improvement greatly simplifies the 3-D object detection framework and reduces the difficulty of deploying the algorithm on edge computing platforms. TANet [18] enhances the pillar features using the triple attention mechanism. PiFeNet [20] proposes a real-time 3-D pedestrian detection network that adopts a pillar-aware attention module for feature enhancement. PillarNet [21] proposes a sparse 2-D convolutional network for spatial feature extraction and high-level and low-level feature fusions. TinyPillarNet [28] designs an extremely tiny pillar-based network for application in the tiny machine learning (TinyML) scenario. Although the pillar-based methods are efficient and detached from 3-D convolution, the 3-D information contained in the point clouds is inevitably lost during the projection from 3-D space to BEV plane. Therefore,

it is more challenging for the pillar-based methods to predict accurate 3-D boxes than the voxel-based methods.

B. Point-Based Methods

Point-based methods [29], [30], [31], [32], [33], [34] directly feed the irregular raw point clouds into the neural network as input. PointNet [35] proposes a multilayer perceptron (MLP) network for pointwise feature aggregation. PointNet++ [36] leverages the farthest point sampling (FPS) and ball-query operation to learn more fine-grained local features from point clouds. F-PointNet [29] is a pioneer that utilizes the point-based network for 3-D object detection in transportation scenarios. PointRCNN [30] classifies the foreground points based on the point features and generates final 3-D boxes with a refinement network. Pointformer [31] employs both local and global transformer-based networks to enhance pointwise feature learning. VoteNet [37] introduces a lightweight Hough voting mechanism to obtain proposals from the centers. The 3DSSD [32] eliminates the need for feature propagation layers by proposing the feature-furthest point sampling (F-FPS) to select the foreground points, which not only improves the efficiency but also reduces the redundant detection boxes. SASA [33] further presents the distance-furthest point sampling (D-FPS) to maintain more foreground points during pointwise feature learning. IA-single-stage detection (SSD) [34] leverages a centroid-aware sampling strategy to improve detection accuracy and implements a lightweight detection framework.

C. Point-Voxel Hybrid Methods

Point-voxel approaches [8], [9], [10], [11], [38], [39], [40] use both points and voxels for 3-D feature representation in hybrid architectures. These methods benefit from the computational efficiency brought by the voxel-based network and the fine-grained 3-D information in point features. HVPR [39] proposes an attentive feature enhancement module in a hybrid single-stage network to improve detection accuracy. More research focuses on two-stage methods for higher performance. Fast Point R-CNN [41] generates 3-D boxes with a voxel-based proposal generation network and extracts the features of each proposal directly from raw point clouds for a lightweight refinement. PV-RCNN [5] proposes a voxel SA layer to aggregate pointwise features from the voxel-wise features. Besides, a fine-grained refinement scheme is adopted in PV-RCNN [5] by dividing the 3-D boxes into several grid points to aggregate features. M3DETR [8] learns multirepresentation features of point clouds and employs a multiscale transformer to fuse these features. PDV [12] uses a density-aware strategy to further develop the point-voxel hybrid pipeline. ARFA [13] proposes an adaptive mechanism to optimize the receptive field for semantic information aggregation, which effectively enhances the detection performance for small objects. SIENet [42] and 3ONet [43] predict the spatial shapes of point clouds in the proposals, and then refine the boxes with the spatial information.

Most point-voxel two-stage methods follow the pipeline that employs a voxel-based detector to obtain the preliminary proposals and further optimizes the proposals through a point-based refinement network. Despite substantial progress, these methods are still limited by the slow inference speed and high computational cost. PP-RCNN [44] designs a two-stage hybrid framework that adopts a pillar-based network for proposal generation and a point-based network for refinement. Although this approach is more efficient, there is still a notable performance gap between PP-RCNN [44] and PV-RCNN [5]. In contrast, our proposed hybrid point-pillar network can better exploit the 3-D spatial information of pointwise features to enhance the accuracy of both the preliminary proposals and final results.

III. Our Method

A. Architecture Overview

In this article, we introduce an innovative hybrid point-pillar network that utilizes both pointwise and pillarwise feature representations. The workflow of our HybridPillars is illustrated in Fig. 1(c). The raw point cloud is not immediately sent to the voxelization. Instead, we utilize a point-based encoder to learn the pointwise features, which preserves the 3-D information as much as possible. The point features with rich 3-D information are subsequently transformed into pillarwise features through a pillar feature aggregation network. The pillar features are used for preliminary 3-D proposal generation, while the point features can provide sufficient 3-D information for proposal refinement. For the second stage, we optimize the 3-D proposals with a point-based refinement network, following the point-voxel two-stage method [5]. It should be noted that as the accurate point features have already been obtained in the first stage, we do not need any additional computation for point feature learning.

The key improvement of our method is that we successfully integrate the acquisition of point features into the hybrid pillar-based proposal generation network. This improvement can provide richer 3-D information for BEV feature learning and proposal refinement, thus boosting the detection performance. Furthermore, this novel architecture is more efficient without the time-consuming point-voxel feature learning.

As illustrated in Fig. 2, the four main components of our HybridPillars are a point feature encoder, a pillar feature aggregation network, a region proposal network (RPN), and a refinement network. In Sections III-B–III-E, we will describe our model in detail.

B. Point Feature Encoder

PointNet++ [36] can learn accurate point features from the irregular point clouds, which has been applied in several point-based methods [29], [30], [31], [32], [33], [34]. An SA block of [36] is employed in our network as the point feature encoder for preliminary point feature learning. The first step of our point feature encoder is to sample N_0 key points using FPS. Next, the neighboring points are clustered into groups by the ball query. The features of each group are then extracted through the MLP block. By setting several radii during the ball query, the multiscale 3-D contextual information is aggregated into the key points. The output point features are of size (N_0, C), where C indicates the channel count for point features.

Motivated by the single-stage point-based methods [33], [34], an auxiliary task is appended at the end of our point feature encoder. This task performs instance segmentation

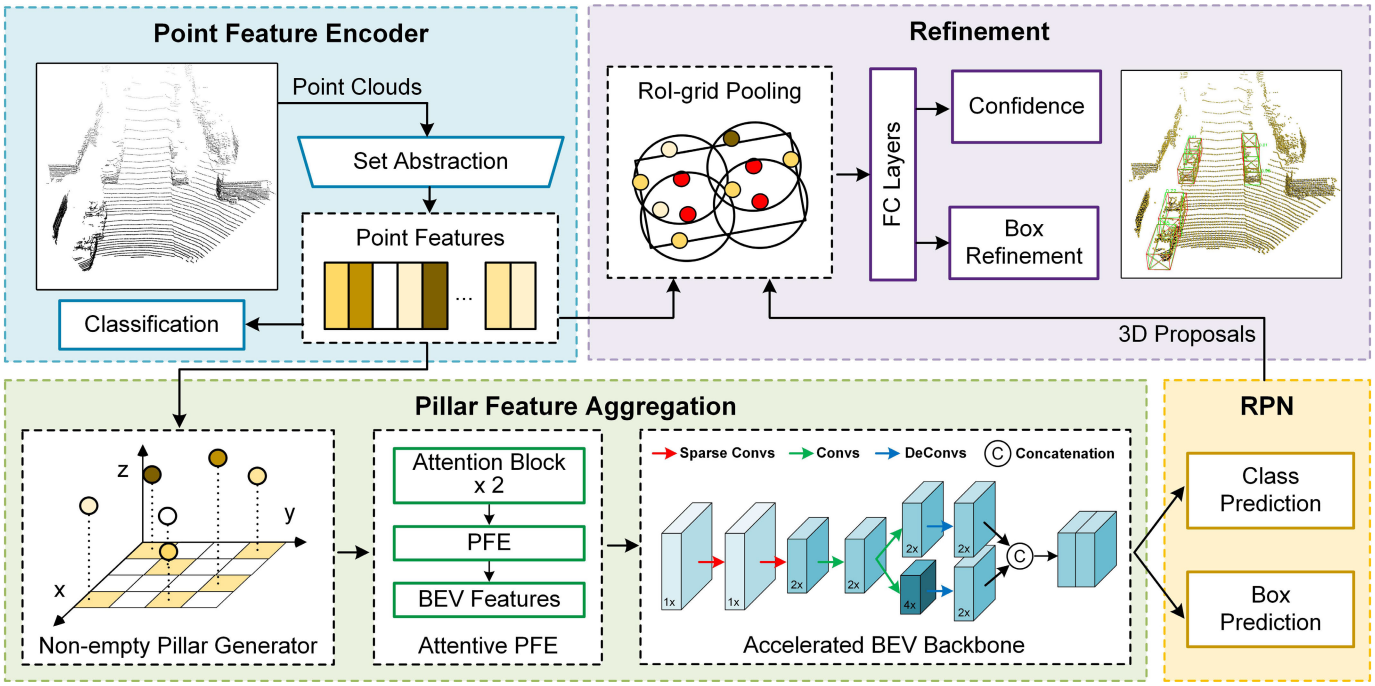


Fig. 2. Architecture of our proposed HybridPillars. The main components of our two-stage framework are a point feature encoder, a pillar feature aggregation network, an RPN, and a refinement network.

for key points and generates scores for the corresponding categories, which can supplement the semantic information of point features. Note that this module is only activated during model training. Therefore, the efficiency of inference computation will not be affected.

In contrast to the raw point clouds, the encoded point features contain richer multiscale information and semantic information, which can benefit both proposal generation and refinement in our hybrid framework. Furthermore, the number of points is reduced by the downsampling operation in the SA block, which contributes to the less computational cost of the subsequent network.

C. Pillar Feature Aggregation

The pillar feature aggregation network is proposed to learn the BEV features from the pointwise features. It consists of three submodules, including a nonempty pillar generator, an attentive PFE, and an accelerated BEV backbone.

1) Nonempty Pillar Generator: Most existing pillar-based methods adopt the voxelization operation of VoxelNet [24] to convert the raw points into pillars before the BEV backbone. However, we find that this operation cannot conduct gradients during experiments. If we use it to project the pointwise features in our HybridPillars, the point feature encoder cannot be trained correctly. Therefore, we propose a gradient conductive pillar generator based on the principle of voxelization. Furthermore, we optimize the voxelization pipeline by computing only the nonempty pillar to reduce the redundant computation.

Given the 3-D space of range H, W, and D along the X-, Y-, and Z-axes, we define each pillar of size (v_H, v_W, D). From Fig. 3, we project the input points to the XY plane and calculate a 2-D occupancy map of size (H', W'), where each pixel indicates the count of points belonging to the

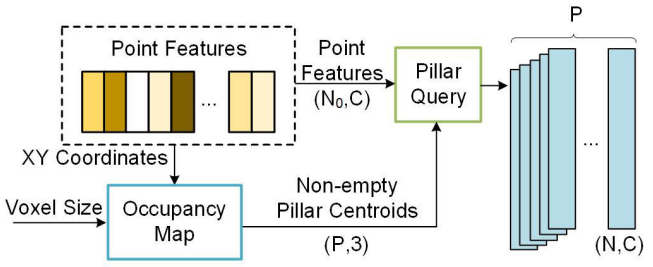


Fig. 3. Nonempty pillar generator. We calculate 2-D occupancy maps to locate nonempty pillars. The pillar query is used to divide the point features into groups around centroids of the nonempty pillars.

corresponding pillar. According to the resulting occupancy map, we can quickly obtain the count of nonempty pillars P and the centroids coordinates of each pillar (C, P, 3). Next, the input features are partitioned into groups around these centroids. To ensure that the gradient can be conducted in this procedure, we design a pillar query operation following the ball query of PointNet++ [36]. Finally, we feed the centroids and the input features to the pillar query operation to generate grouped features. The output features are of size (P, N, C), where N represents the number of points contained in one pillar.

2) Attentive PFE: After identifying the nonempty pillars and clustering the features, we need to learn the features within each pillar based on the collected pointwise features. PFE layer is proposed for pillar feature learning by PointPillars [17], which includes a linear layer to expand the feature channels and a max operation to select the most significant features. To further enhance the representation of pillar features, an attentive PFE layer is proposed motivated by Liu et al. [18] and Le et al. [20]. As shown in Fig. 4, the proposed attention block adopts both pointwise and

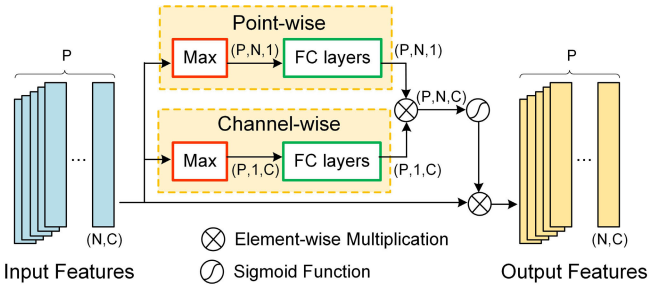


Fig. 4. Architecture of the attention block in HybridPillars. The input features generate an attention matrix through the pointwise and channelwise attention networks.

channelwise attentive enhancements in each group before the PFE. Specifically, we utilize a max operation and several fully connected layers to predict the pointwise attention vectors of size $(P, N, 1)$ and channelwise attention vectors of size $(P, 1, C)$ on the input features. The attention vectors are then fed to an elementwise multiplication to generate a full attention matrix of size (P, N, C). Then, a sigmoid function is employed to normalize the values of this matrix. Ultimately, we multiply the input features with the attention matrix to obtain the attention-weighted features in both channelwise and pointwise dimensions.

To achieve better performance, we stack two attention blocks in our network. The enhanced point features of each pillar are then fed into a PFE layer to learn the pillar features of size (P, C_P), where C_P is the number of channels for pillar features. To facilitate convolutional computation in the following BEV backbone, we place the pillar features back into BEV according to the corresponding pixel coordinates. The obtained BEV features are of size $(H', W'$, and $C_P)$.

3) Accelerated BEV Backbone: The initial BEV features need to be further encoded through a 2-D convolutional network before being used by the RPN for proposal generation. To improve the inference efficiency on GPU platforms, we propose an accelerated BEV backbone. Sparse operation is effective in accelerating convolutional calculation when there are a lot of empty units in the processed features [15]. It should be noted that the BEV features learned from point features in our network are much sparser than those learned from the raw points. Therefore, we introduce sparse 2-D convolution into the BEV backbone to reduce the redundant computation on empty pixels, which can effectively boost the efficiency of our network. From Fig. 2, the initial BEV features first pass through two sparse blocks for feature learning. Each block contains two sparse 2-D convolutional layers. In the second sparse block, we reduce the size of the BEV features to $(H'/2, W'/2)$, which also decreases the sparsity of the features. Note that the sparse layers in our network can be directly replaced by normal 2-D convolutional layers without degrading the accuracy of the network. This modification is typically required when deploying the network to edge computing platforms.

The remaining part of the network is constructed by vanilla 2-D convolutional and deconvolutional blocks. The BEV features are recovered to the size of $(H'/2, W'/2)$ through deconvolutional blocks. A concatenation layer is employed to combine the upsampled BEV features. The output features are fed to the RPN for 3-D proposal generation.

D. Proposal Generation and Refinement Network

A proper RPN is important for a two-stage detection framework. In contrast to the point-based RPN, the BEV-based RPN can generate 3-D proposals with higher recall, which can provide more effective proposals for the refinement network. Therefore, our framework adopts a BEV-based detection head that can generate the 3-D boxes and classes based on the encoded BEV features from the pillar feature aggregation network.

After obtaining sufficient 3-D proposals generated by the RPN, we propose a refinement network to further select and optimize these proposals. Following PV-RCNN [5], we employ a grid-based region of interest (RoI) pooling operation that aggregates pointwise features to grid points of each proposal. Specifically, we uniformly generate $6 \times 6 \times 6$ virtual grid points within each 3-D proposal. Then, we employ PointNet with a ball query to cluster the neighboring point features. To obtain fine-grained and multiscale features for box refinement, we design two groups of PointNet with different ball radii to learn features from raw points and point features, respectively. Note that as we have obtained the accurate point features in the encoder, our method does not need any extra branch for pointwise feature aggregation like the voxel SA of [5]. The learned RoI features of each box are transformed by MLP layers to refine the location and shape of selected 3-D boxes.

E. Loss Function

The proposed HybridPillars can be trained end-to-end with the instance loss L_{ins}, the RPN loss L_{RPN}, and the refinement loss L_{refine}. The total loss is defined as

$$L = L_{\text{RPN}} + L_{\text{refine}} + L_{\text{ins}}. \tag{1}$$

The RPN loss L_{RPN} and the refinement loss L_{refine} are used to supervise the RPN and the refinement network. We adopt the L_{RPN} from anchor-based methods [15], [17]. The definition of refinement loss L_{refine} follows PV-RCNN [5].

The instance loss is calculated on the auxiliary segmentation task. We employ the cross-entropy loss with a soft point mask according to [34]. The definition of instance loss is as follows:

$$L_{\text{ins}} = -\sum_{c=1}^{C}\left(M_i \cdot s_i \log\left(\hat{s_i}\right) + (1 - s_i)\log\left(1 - \hat{s_i}\right)\right) \tag{2}$$

where C denotes the number of categories, M_i is the soft point mask, s_i denotes the instance category labels, and $\hat{s_i}$ is the predicted results.

IV. Experiments

We evaluate our method on the KITTI object detection benchmark [22] and the ONCE dataset [23]. The KITTI benchmark [22] has been widely used for 3-D object detection evaluation. It provides 7481 frames for training and 7518 frames for testing. Following the common setting [5], [15], [17], we further subdivide the training frames into a

TABLE I
PERFORMANCE COMPARISON OF 3-D OBJECT DETECTION WITH EXISTING METHODS ON KITTI TEST SPLIT

	Methods	Type	Car 3D AP (%)			Car BEV AP (%)			Cyc. 3D AP (%)			Cyc. BEV AP (%)		
			Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
1-stage	VoxelNet [24]	V	77.47	65.11	57.73	89.35	79.26	77.39	61.22	48.36	44.37	66.70	54.76	50.55
	SECOND [15]	V	84.65	75.96	68.71	91.81	86.37	81.04	-	-	-	-	-	-
	PointPillars [17]	B	82.58	74.31	68.99	90.07	86.56	82.81	77.01	58.65	51.92	79.90	62.73	55.58
	MVDCANet [45]	B	88.01	78.58	75.49	93.44	89.14	87.21	84.65	64.12	59.97	89.40	68.74	64.46
	3DSSD [32]	P	88.36	79.57	74.55	92.66	89.02	85.86	82.48	64.10	56.90	85.04	67.62	61.14
	IA-SSD [34]	P	88.34	80.13	75.04	92.79	89.33	84.35	78.35	61.94	55.70	81.30	66.29	59.58
2-stage	PointRCNN [30]	P	86.96	75.64	70.70	92.13	87.39	82.72	74.96	58.82	52.53	82.56	67.24	60.28
	PP-RCNN [44]	P+B	87.31	80.56	76.03	91.90	87.80	85.14	72.75	57.59	52.03	74.30	60.87	55.45
	PV-RCNN [5]	P+V	90.25	81.43	76.82	94.98	90.65	86.14	78.60	63.71	57.65	82.49	68.89	62.41
	PDV [12]	P+V	90.43	81.86	77.36	94.56	90.48	86.23	83.04	67.81	60.46	85.54	71.31	64.4
	ARFA [13]	P+V	90.1	82.05	78.19	-	-	-	85.28	69.72	62.12	-	-	-
	HybridPillars	P+B	90.06	81.48	76.94	94.65	90.34	86.08	81.42	66.05	59.59	83.02	70.19	62.80

training set (3712 frames) and a validation set (3769 frames) for local evaluation. The evaluation metric is average precision (AP) with a 40 recall rate. ONCE dataset [23] is a large autonomous driving dataset for 3-D object detection which provides 16k fully annotated LiDAR scenes with 3-D bounding boxes. We follow the official split set to train and evaluate the models on the ONCE dataset.

A. Implementation Details

We implement HybridPillars based on the OpenPCDet framework [46]. To align the BEV features with the point clouds, we clip the range of points in Cartesian coordinates into [0.0, 69.12] m for the X-axis, [−39.68, 39.68] m for the Y-axis, and [−3.0, 1.0] m for the Z-axis, respectively. Then, we filter the raw point clouds through a projection from 3-D space to the perspective of the camera. The remaining points are sampled to 16 384 points as the input for our network. The point feature encoder is configured to utilize 4096 key points, which is one-fourth of the input point clouds. For the ONCE dataset, we modify the range of point clouds to [−75.2, 75.2] m for the X-axis, [−75.2, 75.2] m for the Y-axis, and [−5.0, 3.0] m for the Z-axis, respectively. The number of key points increases to 30 000. For each key point, we perform three ball query operations with radii of 0.05, 0.2, and 0.8 m to aggregate 3-D features of neighboring point clouds.

The voxel sizes for the pillar generator are configured to (0.16 m, 0.16 m, 4 m) and the feature dimensions of the attentive PFE are set to 64. The BEV backbone consists of four 2-D convolutional blocks and two deconvolutional blocks. The first two convolutional blocks reduce the scale of BEV features to one-half and expand the feature dimensions from 64 to 128, which can be further accelerated by sparse convolution. The remaining blocks are constructed using vanilla 2-D convolutional and deconvolutional layers to further encode the BEV features. The final features with 256 channels are obtained through a concatenation operation at the end of the BEV backbone. The grid resolution of the pooling operation in the refinement network is set to $6 \times 6 \times 6$, which means that 216 virtual grid points are generated for each 3-D box. Then, we utilize PointNet to aggregate RoI features from both the raw point clouds and the learned point features. The radii of the ball query are (0.05 m, 0.2 m) and (0.8 m, 1.6 m) for raw point clouds and point features aggregation, respectively. During the ball query operation, we randomly select 16 points neighboring the virtual grid points. The aggregated features are concatenated and transformed into the feature vectors with 256 channels for the final refinement of each proposal.

HybridPillars is trained with Adam optimizer [47] and one cycle learning rate schedule. We use 0.01 as the initial learning rate. Moreover, we adopt three data augmentation strategies, including random flipping, random scaling, and random rotation. We train the model for 80 epochs on the KITTI dataset and ONCE dataset.

B. Results on KITTI Dataset

We evaluate the performance of our model on the KITTI test dataset and compare our results with several methods from existing literature. The 3-D AP and BEV AP scores of car and cyclist detection are reported in Table I. According to the feature representation of 3-D information in the network, we classify the methods into several types including voxel-based methods (V), point-based methods (P), and pillar/BEV-based methods (B). Some hybrid networks utilize more than one kind of feature representation in a single network, such as point-voxel methods (P + V) and point-pillar/BEV methods (P + B).

We report the performance of both single-stage and two-stage methods in Table I. From the top part, it can be observed that the point-based networks, IA-SSD [34] and 3DSSD [32], achieve higher performance on 3-D AP of car category than other one-stage detectors, which illustrates the strong capability of the point-based feature encoding network. However, there is still a performance gap between one-stage and two-stage networks. The performance of several mainstream two-stage methods and our HybridPillars is reported in the bottom part of Table I. It is shown that the 3-D AP on moderate car detection of two-stage P + B methods, PP-RCNN [44] and HybridPillars, outperforms that of IA-SSD [34] by 0.43% and 1.35%, respectively. Compared with the high-performance P + V methods, our HybridPillars outperforms PV-RCNN [5] but lags behind PDV [12] and ARFA [13]. This issue is probably because the voxelization of pillar-based network losses more 3-D information, which also occurs on PointPillars [17] and SECOND [15]. Overall, the aforementioned experimental results indicate that our method outperforms all P + B methods. Moreover, our method is

TABLE II
PERFORMANCE COMPARISON ON ONCE VALIDATION SET

Methods	Vehicle 3D AP (%)				Cyclist 3D AP (%)				mAP
	overall	0-30m	30-50m	>50m	overall	0-30m	30-50m	>50m	
SECOND [15]	71.19	84.04	63.02	47.25	58.04	69.96	52.43	34.61	64.62
PointPillars [17]	68.57	80.86	62.07	47.04	46.81	58.33	40.32	25.86	57.69
CenterPoint [48]	66.79	80.10	59.55	43.39	63.45	74.28	57.94	41.48	65.12
IA-SSD [34]	70.30	83.01	62.84	47.01	62.17	73.78	56.31	39.53	66.24
PointRCNN [30]	52.09	74.45	40.89	16.81	29.84	46.03	20.94	5.46	40.97
PV-RCNN [5]	77.77	89.39	72.55	58.64	59.37	71.66	52.58	36.17	68.57
HybridPillars	76.60	88.07	71.65	56.04	60.25	71.89	53.22	36.53	68.43

TABLE III
RUNTIME COMPARISON RESULTS WITH OTHER TWO-STAGE METHODS

Method	Type	Backbone	AP_{Car} (%)	Speed (ms)	Params (M)	FLOPs (G)
PointRCNN [30]	P	1D	75.64	100	**4.0**	**27.3**
PP-RCNN [30]	P+B	1D&2D	80.56	67	-	-
PV-RCNN [5]	P+V	2D&3D	81.43	101.2	13.1	92.4
PDV [12]	P+V	2D&3D	81.86	135	12.8	90.3
ARFA [13]	P+V	2D&3D	**82.05**	280	21.1	-
HybridPillars	P+B	1D&2D	81.48	**45.8**	7.1	31.7

TABLE IV
ABLATION STUDIES FOR POINT FEATURE ENCODER ON KITTI VAL SPLIT. THE 3-D AP OF CAR DETECTION IS REPORTED

Method	Input		Module		RCNN	Car 3D AP (%)		
	PC	PF	MS	AT		Easy	Moderate	Hard
HybridPillars (SSD)	✓					**89.33**	78.59	75.52
		✓				89.13	79.71	76.86
		✓	✓			88.45	81.21	78.66
		✓	✓	✓		89.17	**81.60**	**78.84**
HybridPillars	✓					88.84	79.18	76.49
		✓			✓	92.46	84.32	82.24
		✓	✓			92.43	84.99	82.61
		✓	✓	✓		**92.88**	**85.11**	**82.73**

also competitive when compared to the high-performance two-stage P + V methods.

C. Results on ONCE Dataset

To demonstrate the generality and effectiveness of our proposed HybridPillars, we evaluate our method on the ONCE dataset. Table II reports the 3-D AP of several baselines and our detector for vehicle and cyclist categories. It is shown that HybridPillars outperforms all single-stage methods, including SECOND [15], PointPillars [17], CenterPoint [48], and IA-SSD [34]. Compared with the two-stage methods, our network outperforms PointRCNN [30] and slightly lags behind PV-RCNN [5]. This result indicates that our method achieves competitive performance on ONCE dataset, which verifies the effectiveness of our proposed framework.

D. Runtime Analysis

To further analyze runtime performance, we report the network type, backbone type, 3-D AP on moderate car, inference time, the number of parameters, and floating point operations (FLOPs) of the two-stage methods in Table III. Note that the reported statistics of speed are averaged from 3769 runs of our inference program on a computer with one Intel i7-7820X CPU and one TITAN Xp GPU. As shown in Table III, ARFA [13] achieves the highest accuracy but the inference costs 280 ms per frame, which is too slow for real-time application. The accuracy of our HybridPillars on moderate car detection is 81.48%, which is comparable to that of PV-RCNN [5]. Notably, our method only requires 45.8 ms of inference time, which is much faster than other two-stage methods. This result shows that our proposed method boosts the inference speed of the two-stage pipeline while keeping competitive detection performance. In addition, we compare the type of operation in the backbone of these methods. Our proposed method does not use the voxel representation as well as 3-D convolutional layers, which facilitates the deployment on edge computing devices.

The number of parameters and FLOPs are reported in Table III. Note that PointRCNN [30] requires the fewest parameters and operations in the network. This may be due to the fact that PointRCNN employs a 1-D convolutional layers in the point-based backbone. However, its point-based pipeline generates too much proposals in the first stage, which influences the overall speed. Our network requires 7.1M parameters and 31.7G FLOPs, both of which are fewer than other hybrid two-stage methods. This result demonstrates the efficiency of our HybridPillars.

E. Qualitative Results

We visualize qualitative results in Fig. 5, where the detection boxes in green and the boxes of ground truth in red. It can be observed that the two-stage PV-RCNN [5] and our HybridPillars perform better than the single-stage PointPillars [17]. For example, in the first row of Fig. 5, four cars on the far right are missed by the single-stage PointPillars, while PV-RCNN and our method can identify some of them.

F. Ablation Study

In this section, we investigate the components and the variants of HybridPillars with ablation experiments. We train the model in consistency with the training settings on the KITTI train split and evaluate our model using 3-D AP on val split. We report the performance of the car category, which contains the most training samples.

1) Effects of Point Feature Encoder: We first investigate the effect of the point feature encoder in our HybridPillars. For comparison, we report the performance of using raw point clouds as the input to the pillar feature aggregation network. As shown in Table IV, PC represents raw point

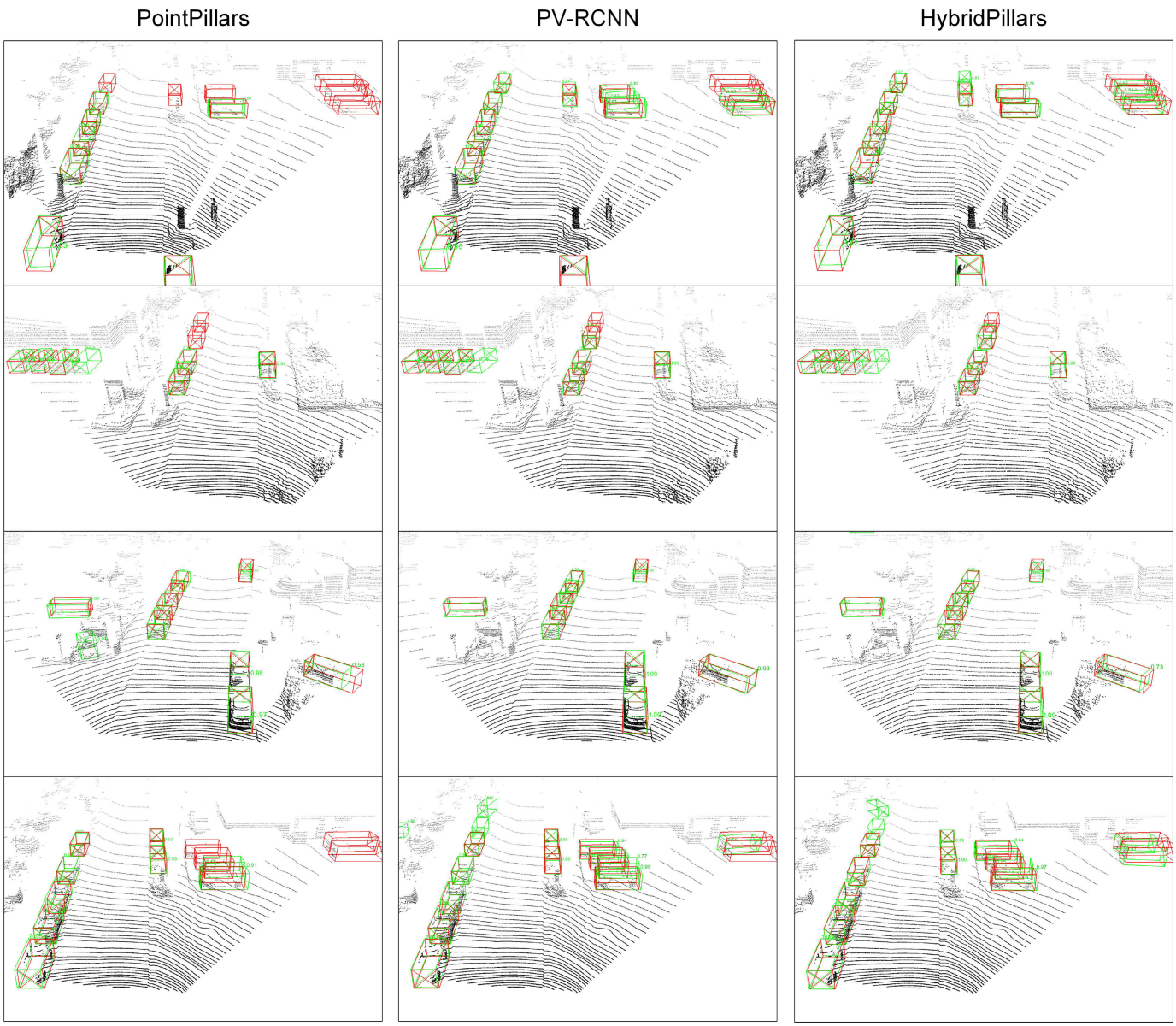


Fig. 5. Qualitative results of PointPillars, PV-RCNN, and our method on the KITTI benchmark.

clouds while PF means the point features. Compared with using raw point clouds as input, we observe that using point features can significantly enhance the detection performance. The single-stage model using point features yields the 3-D AP by 79.71% on moderate difficulty, outperforming the model with raw point cloud by 1.12%. To verify the superiority of point features, we also provide the performance of two-stage models, which utilize the raw point clouds and point features for proposal refinement through the RCNN. From the fifth row and the sixth row of Table IV, we observe that the accuracy of 3-D proposals can be further boosted by 3.51% through the point features while relying on the raw point clouds slightly improves the performance by 0.59% on moderate car detection. This result indicates that the encoded features can provide richer 3-D contextual information than the raw points, which is effective in improving the performance of proposal generation and refinement. Moreover, we also investigate the effect of multiscale feature aggregation and the auxiliary classification task in our point feature encoder, which are named MS and AT in Table IV. The multiscale strategy boosts the 3-D AP on moderate car detection by 1.5% for the single-stage model and 0.67% for the two-stage model. The two-stage model with auxiliary task achieves the improvement by 0.45%, 0.12%, and 0.12% on three difficulties, respectively. The above results verify the effectiveness of applying the multiscale feature aggregation and auxiliary classification task.

2) Effects of Nonempty Pillar Generator: As mentioned in Fig. 3, our pillar generator is composed of two main procedures, including occupancy map calculation and pillar query. Table V measures the time consumption of normal pillarization and our nonempty pillar generator with different input features, where OM represents the occupancy map calculation and PQ represents the pillar query. We set up four groups of point features with 64 channels as input, which are of sizes

TABLE V
TIME CONSUMPTION OF NORMAL PILLARIZATION AND OUR NONEMPTY PILLAR GENERATOR WITH DIFFERENT SIZES OF INPUT FEATURES

Size of Point Features (C=64)	Normal Pillarization (size=0.16)	Non-empty Pillar Generator (size=0.16)		
		OM	PQ	Total
16384	12.1 ms	0.8 ms	3.5 ms	4.3 ms
8192	9.9 ms	0.8 ms	2.0 ms	2.8 ms
4096	8.9 ms	0.7 ms	1.1 ms	1.8 ms
1024	7.3 ms	0.7 ms	0.6 ms	1.3 ms
Gradient Conductivity	No	Yes		

TABLE VI
EFFECTS OF THE PILLAR SIZES ON PERFORMANCE AND INFERENCE SPEED

Method	Pillar Size (m)	Car 3D AP (%)			Time (ms)
		Easy	Moderate	Hard	
PointPillars	0.16	87.75	78.39	75.18	28.3
HybridPillars (SSD)	0.12	88.88	79.91	77.12	33.2
	0.16	88.61	79.70	76.84	23.5
	0.20	88.90	79.54	76.57	18.3
	0.24	88.49	79.22	76.48	15.1

16 384, 8192, 4096, and 1024. The pillar size is uniformly set to (0.16 m, 0.16 m) along the X- and Y-axes. It can be observed that the processing time of normal pillarization is contingent upon the size of the input features. When fed with point features of size 16 384, it consumes 12.1 ms to complete the calculation. In comparison, our pillar generator significantly reduces the computation time. As shown in Table V, the time consumed by PQ is also positively related to the size of the input features, while the computation time of OM remains between 0.7 and 0.8 ms. When processing 16 384 points, our pillar generator takes a total of 4.3 ms, of which 0.8 ms is used for calculating the occupancy map and 3.5 ms is used for the pillar query. In other words, our method accelerates about 64% over the traditional pillarization in this case. It should be noted that the fewer the point features are, the better the acceleration of our method is. Our method accelerates at 64%, 72%, 80%, and 82% for the input of sizes 16 384, 8192, 4096, and 1024, respectively. Additionally, our implemented pillar generator can conduct the gradient, which is essential for the conversion from point features to pillar features.

3) Effects of Pillar Size: To study the pillar-based network in our hybrid framework, we investigate the effect of different pillar sizes in our single-stage model. From Table VI, we report the detection accuracy and inference speed of the models with four different pillar sizes. For comparison, we also report the performance and speed of PointPillars [17], which is reproduced in the OpenPCDet framework [46]. It can be seen that our model outperforms PointPillars in both accuracy and speed with the pillar size of 0.16 m. Even when enlarging the pillar size to 0.24 m, the accuracy of our model is still better than PointPillars. These results demonstrate the effectiveness of our hybrid framework. It should be noted that the models involved in other experiments use the pillar size of 0.16 m, which is consistent with PointPillars [17].

4) Effects of Attention Block: Next, we investigate the effects of attention block in our pillar feature aggregation network.

TABLE VII
EFFECTS OF THE ATTENTIVE PFE. THE EXPERIMENTS ARE BASED ON THE SINGLE-STAGE MODEL HYBRIDPILLARS (SSD)

Attention Module		Car 3D AP (%)		
Point-wise	Channel-wise	Easy	Moderate	Hard
		88.75	79.67	76.89
✓		89.04	79.89	77.94
	✓	89.09	79.60	77.60
✓	✓	**89.17**	**81.60**	**78.84**

TABLE VIII
ABLATION STUDIES FOR ROI-GRID POOLING OPERATION IN THE REFINEMENT NETWORK

Grid Size	Car 3D AP (%)			Time (ms)
	Easy	Moderate	Hard	
$4 \times 4 \times 4$	92.45	84.76	82.31	36.5
$5 \times 5 \times 5$	92.44	85.09	82.57	39.6
$6 \times 6 \times 6$	92.55	85.11	82.63	45.8
$7 \times 7 \times 7$	92.61	85.16	82.58	53.9
$8 \times 8 \times 8$	92.42	85.21	82.76	64.5

Since the enhanced pillar features are mainly used to generate detection proposals in the first stage of the detection network, we report the effect of different attention strategies on HybridPillars (SSD) in Table VII. The second and third rows show the performance of using pointwise or channelwise attention. It can be observed that the pointwise attention module enhances the 3-D AP by 0.29%, 0.22%, and 1.05% on three difficulties, respectively. When performing both pointwise and channelwise attention on pillar features, the model achieves 89.17%, 81.60%, and 78.84% accuracy on three difficulties, outperforming the baseline by 0.42%, 1.93%, and 1.95%. The above experimental results suggest that the proposed attention block can effectively aggregate 3-D information into the pillar features, thus improving the performance of our method.

5) Resolution of Grid-Based RoI Pooling: Our HybridPillars employs a grid-based RoI pooling module for proposal refinement. From Table VIII, we compare the inference time and the detection performance of different grid sizes in the refinement network to investigate the effect of this module. The results show that with higher resolution of grids, the detection performance of our model is better, but the inference speed is also slower. Therefore, we adopt the grid size of $6 \times 6 \times 6$ to balance the speed and the performance of the refinement network, which is identical to PV-RCNN [5].

6) Effects of Sparse Convolutional Layers and Model Deployment: Our BEV backbone utilizes several sparse convolutional layers to extract dense feature maps from input sparse features. The main advantage is that it can accelerate the inference of the network by exploiting the sparsity of the input features. We compare the inference time and detection accuracy of using sparse convolution and normal convolution, named Sparse and Dense in Table IX. It is shown that the sparse convolutional layers can reduce the inference time by about 12.7 ms compared with normal convolution. However, despite being efficient, the sparse convolution is not supported by many deployment frameworks currently due to its computation complexity. Considering this issue, we propose to replace

TABLE IX
EFFECTS OF THE SPARSE CONVOLUTIONAL LAYERS IN OUR BEV BACKBONE. BOTH THE 3-D AP AND INFERENCE TIME ARE REPORTED

2D Backbone		TensorRT	Car 3D AP (%)			Time
Sparse	Dense	deployment	Easy	Moderate	Hard	(ms)
✓			92.55	85.11	82.63	45.8
	✓		92.60	85.09	82.63	58.5
	✓	✓	92.59	85.08	82.62	38.2

the sparse convolution with normal convolution before model deployment. As shown in Table IX, this kind of replacement will not degrade the performance of our network. Then, we further deploy our model with TensorRT, which is the most commonly used framework for artificial intelligence deployment on NVIDIA edge computing platform. Our backbone can be easily converted to a TensorRT engine since all operators are now naturally supported. During deployment, we use FP16 as the target precision of the deployed model, which can effectively avoid the degradation of detection performance. As shown in Table IX, the inference speed of the deployed model can be reduced to 38.2 ms, which is 20.3 and 7.6 ms faster than the dense model and the sparse model, respectively. This easy-to-implement porting scheme illustrates that our proposed method is friendly for deployment.

V. CONCLUSION

We have proposed a novel two-stage hybrid point-pillar network, named HybridPillars, for real-time and deployment-friendly 3-D object detection. We first propose an optimized framework that integrates the acquisition of point features into a hybrid point-pillar network. Compared with existing pillar-based approaches, our method provides richer 3-D spatial information in the point features for proposal generation and refinement. Then, we present a pillar feature aggregation network consisting of a nonempty pillar generator, an attentive PFE module, and an accelerated BEV backbone. This network not only converts point features to BEV features efficiently but also improves the performance of our detector through the attentive feature enhancement. Extensive experiments verify the efficiency and effectiveness of HybridPillars compared with other two-stage point-voxel detectors. Moreover, the experiment of model deployment demonstrates that our proposed model can be easily ported to the deployment framework, which is important for the real-time LiDAR-based applications.

REFERENCES

[1] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3D point clouds: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 12, pp. 4338–4364, Dec. 2020.

[2] E. Arnold, O. Y. Al-Jarrah, M. Dianati, S. Fallah, D. Oxtoby, and A. Mouzakitis, "A survey on 3D object detection methods for autonomous driving applications," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 10, pp. 3782–3795, Oct. 2019.

[3] Y. Cui et al., "Deep learning for image and point cloud fusion in autonomous driving: A review," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 2, pp. 722–739, Feb. 2021.

[4] J. Mao, S. Shi, X. Wang, and H. Li, "3D object detection for autonomous driving: A comprehensive survey," 2022, *arXiv:2206.09474*.

[5] S. Shi et al., "PV-RCNN: Point-voxel feature set abstraction for 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10529–10538.

[6] J. Mao, M. Niu, H. Bai, X. Liang, H. Xu, and C. Xu, "Pyramid R-CNN: Towards better performance and adaptability for 3D object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 2723–2732.

[7] H. Shenga et al., "Improving 3D object detection with channel-wise transformer," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 2743–2752.

[8] T. Guan et al., "M3DETR: Multi-representation, multi-scale, mutual-relation 3D object detection with transformers," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2022, pp. 2293–2303.

[9] J. Li et al., "P2V-RCNN: Point to voxel feature learning for 3D object detection from point clouds," *IEEE Access*, vol. 9, pp. 98249–98260, 2021.

[10] K. Ning, Y. Liu, Y. Su, and K. Jiang, "Point-voxel and bird-eye-view representation aggregation network for single stage 3D object detection," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 3, pp. 3223–3235, Mar. 2023.

[11] G. Shi, K. Wang, R. Li, and C. Ma, "Real-time point cloud object detection via voxel-point geometry abstraction," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 6, pp. 5971–5982, Jun. 2023.

[12] J. S. K. Hu, T. Kuai, and S. L. Waslander, "Point density-aware voxels for LiDAR 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 8459–8468.

[13] D. Zhang, X. Wang, Z. Zheng, X. Liu, and G. Fang, "ARFA: Adaptive reception field aggregation for 3D detection from LiDAR point cloud," *IEEE Sensors J.*, vol. 23, no. 11, pp. 11156–11167, Nov. 2022.

[14] J. Deng, S. Shi, P. Li, W. Zhou, Y. Zhang, and H. Li, "Voxel R-CNN: Towards high performance voxel-based 3D object detection," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 2, May 2021, pp. 1201–1209.

[15] Y. Yan, Y. Mao, and B. Li, "SECOND: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, Oct. 2018.

[16] S. Zhou et al., "Fastpillars: A deployment-friendly pillar-based 3D detector," 2302, *arXiv:2302.02367*.

[17] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast encoders for object detection from point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12697–12705.

[18] Z. Liu, X. Zhao, T. Huang, R. Hu, Y. Zhou, and X. Bai, "TANET: Robust 3D object detection from point clouds with triple attention," in *Proc. Conf. Artif. Intell. (AAAI)*, Feb. 2020, pp. 11677–11684.

[19] Y. Wang et al., "Pillar-based object detection for autonomous driving," in *Proc. 16th Eur. Conf. Comput. Vision (ECCV)*, Glasgow, U.K. Cham, Switzerland: Springer, 2020, pp. 18–34.

[20] D. T. Le, H. Shi, H. Rezatofighi, and J. Cai, "Accurate and real-time 3D pedestrian detection using an efficient attentive pillar network," *IEEE Robot. Autom. Lett.*, vol. 8, no. 2, pp. 1159–1166, Feb. 2023.

[21] G. Shi, R. Li, and C. Ma, "Pillarnet: Real-time and high-performance pillar-based 3D object detection," in *Proc. 17th Eur. Conf. Comput. Vision (ECCV)*, Tel Aviv, Israel. Cham, Switzerland: Springer, 2022, pp. 35–52.

[22] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.

[23] J. Mao et al., "One million scenes for autonomous driving: ONCE dataset," 2021, *arXiv:2106.11037*.

[24] Y. Zhou and O. Tuzel, "VoxelNet: End-to-end learning for point cloud based 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4490–4499.

[25] Q. Xu, Y. Zhong, and U. Neumann, "Behind the curtain: Learning occluded shapes for 3D object detection," in *Proc. AAAI Conf. Artif. Intell.*, 2022, vol. 36, no. 3, pp. 2893–2901.

[26] Y. Chen, J. Liu, X. Zhang, X. Qi, and J. Jia, "VoxelNeXt: Fully sparse VoxelNet for 3D object detection and tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 21674–21683.

[27] G. Zhang, L. Fan, C. He, Z. Lei, Z. Zhang, and L. Zhang, "Voxel Mamba: Group-free state space models for point cloud based 3D object detection," 2024, *arXiv:2406.10700*.

[28] Y. Li, Y. Zhang, and R. Lai, "TinyPillarNet: Tiny pillar-based network for 3D point cloud object detection at edge," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 34, no. 3, pp. 1772–1785, Mar. 2024.

[29] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3D object detection from RGB-D data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 918–927.

[30] S. Shi, X. Wang, and H. Li, "PointRCNN: 3D object proposal generation and detection from point cloud," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 770–779.

[31] X. Pan, Z. Xia, S. Song, L. E. Li, and G. Huang, "3D object detection with pointformer," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 7463–7472.

[32] Z. Yang, Y. Sun, S. Liu, and J. Jia, "3DSSD: Point-based 3D single stage object detector," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11040–11048.

[33] C. Chen, Z. Chen, J. Zhang, and D. Tao, "SASA: Semantics-augmented set abstraction for point-based 3D object detection," in *Proc. AAAI Conf. Artif. Intell.*, vol. 36, no. 1, 2022, pp. 221–229.

[34] Y. Zhang, Q. Hu, G. Xu, Y. Ma, J. Wan, and Y. Guo, "Not all points are equal: Learning highly efficient point-based detectors for 3D LiDAR point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2022, pp. 18953–18962.

[35] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 652–660.

[36] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 5105–5114.

[37] Z. Ding, X. Han, and M. Niethammer, "VoteNet: A deep learning label fusion method for multi-atlas segmentation," in *Proc. 22nd Int. Conf. Med. Image Comput. Comput. Assist. Intervention (MICCAI)*, Shenzhen, China. Cham, Switzerland: Springer, Oct. 2019, pp. 202–210.

[38] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia, "STD: Sparse-to-dense 3D object detector for point cloud," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1951–1960.

[39] J. Noh, S. Lee, and B. Ham, "HVPR: Hybrid voxel-point representation for single-stage 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 14605–14614.

[40] Y. Li, S. Yang, Y. Zheng, and H. Lu, "Improved point-voxel region convolutional neural network: 3D object detectors for autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 9311–9317, Jul. 2022.

[41] Y. Chen, S. Liu, X. Shen, and J. Jia, "Fast point R-CNN," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9775–9784.

[42] Z. Li, Y. Yao, Z. Quan, J. Xie, and W. Yang, "Spatial information enhancement network for 3D object detection from point cloud," *Pattern Recognit.*, vol. 128, Aug. 2022, Art. no. 108684.

[43] H. Anh Hoang and M. Yoo, "3ONet: 3-D detector for occluded object under obstructed conditions," *IEEE Sensors J.*, vol. 23, no. 16, pp. 18879–18892, Jun. 2023.

[44] J. Tu, P. Wang, and F. Liu, "PP-RCNN: Point-pillars feature set abstraction for 3D real-time object detection," in *Proc. Int. Joint Conf. Neural Netw.*, Jul. 2021, pp. 1–8.

[45] X. Gao and D. Hu, "MVDCANet: An end-to-end self-attention-based multiview-dualchannel 3D object detection," *IEEE Sensors J.*, vol. 21, no. 24, pp. 27789–27800, Dec. 2021.

[46] O. D. Team. (2020). *Openpcdet: An Open-source Toolbox for 3D Object Detection From Point Clouds*. [Online]. Available: https://github.com/open-mmlab/OpenPCDet

[47] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," in *Proc. Artif. Intell. Mach. Learn. Multi-Domain Oper. Appl.*, Baltimore, MD, USA, vol. 11006. Bellingham, WA, USA: SPIE, Apr. 2019, pp. 369–386.

[48] T. Yin, X. Zhou, and P. Krahenbuhl, "Center-based 3D object detection and tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 11784–11793.

Zhicong Huang received the B.S. and M.S. degrees from Sun Yat-sen University, Guangzhou, China, in 2019 and 2021, respectively, where he is currently pursuing the Ph.D. degree with the School of Electronics and Information Technology.

His research interests include artificial intelligence and hardware accelerator, especially for autonomous driving.

Yuxiao Huang is currently pursuing the M.E. degree with Sun Yat-sen University, Guangzhou, China.

His main research interests include pillar-based 3-D object detection and neural network accelerator engine.

Zhijie Zheng received the B.S. degree from Sun Yat-sen University, Guangzhou, China, in 2022, where he is currently pursuing the M.E. degree.

His main research interests include computer vision and deep learning, especially for 3-D object detection, multidataset fusion, and multilabel image classification.

Haifeng Hu (Member, IEEE) received the Ph.D. degree from Sun Yat-sen University, Guangzhou, China, in 2004.

He is a Professor with the School of Electronics and Information Technology, Sun Yat-sen University. He has authored about 260 articles since 2000. His research interests are in computer vision, pattern recognition, natural language processing, image processing, and neural computation.

Dihu Chen (Member, IEEE) received the B.S. and M.S. degrees in semiconductor physics from Sichuan University, Chengdu, China, in 1986 and 1989, respectively, and the Ph.D. degree in solid-state electron from the Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong, in 2000.

Since 1989, he has been with Sun Yat-sen University, China. He is currently working on electronic devices, integrated circuit design, and design methodology.