Multiview Attention for 3D Object Detection in Lidar Point Cloud

Kevin Tirta Wijaya*
The Robotics Program
KAIST
Daejeon, Republic of Korea
kevin.tirta@kaist.ac.kr

Donghee Paek*
The Cho Chun Shik
Graduate School of Green Transportation
KAIST
Daejeon, Republic of Korea
donghee.paek@kaist.ac.kr

Seung-Hyun Kong †
The Cho Chun Shik
Graduate School of Green Transportation
KAIST
Daejeon, Republic of Korea
skong@kaist.ac.kr

***Abstract*—Prior works in voxel-based Lidar 3D object detection have demonstrated promising results in detecting a variety of road objects such as cars, pedestrians, and cyclists. However, these works generally reduce the feature space from a 3D volume into a 2D bird eye view (BEV) map before generating object proposals to speed up the inference runtime. As a result, the resolution of information in the z-axis is reduced significantly. In this work, we hypothesize that augmenting the BEV features with features obtained from a front view (FV) map may provide a way for the network to partially recover the high-resolution z-axis information. The augmentation allows object proposals to be inferred in the BEV, maintaining the fast runtime, and simultaneously improving the 3D detection performance. To support our hypothesis, we design a multi-view attention module that augments the BEV features with the FV features and conduct extensive experiments on the widely used KITTI dataset. Based on the experimental results, our method successfully improves various existing voxel-based 3D object detection networks by a significant margin.**

***Index Terms*—3D object detection, Lidar point cloud, multi-view attention**

I. Introduction

Object detection is one of the most researched topics in the field of computer vision. Various 2D object detection networks with remarkable performance have been introduced, owing to the advancements in deep learning, convolutional neural networks (CNN), and transformers. However, 2D object detection alone is often not sufficient for a machine, such as an autonomous car, to operate in a real-world scenario. Objects in the real world exist in the three-dimensional space, therefore, 3D information is required to plan safe maneuvers. Hence, a robust 3D object detection technique is a crucial function for an autonomous car.

3D object detection is often performed in a point cloud obtained from a Lidar sensor. Recent deep learning-based 3D object detection networks for Lidar point cloud [13]–[17] often consist of a preprocessing module, a feature extraction backbone, a detection head, and in the case of two-stage detectors, an additional refinement head.

As Lidar measurements are often stored in an unordered list, the preprocessing module needs to introduce spatial structure to the data so that convolution operations can be applied. The most commonly used preprocessing method is voxelization, where the 3D point cloud space is discretized into non-overlapping cuboids of identical sizes. The attributes of points that lie in the same voxel are used as the feature vectors of the voxel. A series of 3D convolutions are then applied to the voxels, decreasing the spatial dimension of the voxel volume from $\mathbb{R}^{D\times W\times H}$ to $\mathbb{R}^{D'\times W'\times 1}$ where D, W, and H are the voxel volume's depth (x-axis), width (y-axis), and height (z-axis), respectively. This feature extraction process can be seen as a dimensionality reduction process from a 3D feature volume to a 2D bird eye view (BEV) feature map.

There are at least two compelling reasons to choose a 2D BEV feature map as the final feature map. Firstly, object proposals are predicted by applying a shared multilayer perceptron (shared-MLP) to every feature vector in the final feature space, either a 2D feature map or a 3D feature volume. Consequently, performing such a process in the 3D space requires a lot of computations, resulting in prohibitively slow inferences. Secondly, the 2D BEV feature map is an excellent choice of representation since objects of interest on the road (i.e., Cars, Pedestrians, and Cyclists) do not overlap in the BEV. This is not the case for the front-view map, where objects may fall in a line such that farther objects are occluded by nearer objects.

Although the 2D BEV feature map has desirable properties in terms of inference speed and objects separations, the resolution of information in the z-axis is significantly reduced. We hypothesize that the accuracy of 3D bounding box proposals generated from features with low-resolution z-axis information should be lower compared to when high-resolution z-axis information is available.

In order to provide high-resolution information z-axis information to the detection head while maintaining the 2D BEV feature map shape, we propose a new front-view (FV) to BEV feature augmentation via the multiview attention (MVA) module. The MVA module consists of learnable functions that robustly augment the BEV feature vectors with the FV feature vectors that have high-resolution z-axis information at similar

* co-authors, † corresponding author
This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (2021R1A2C300837011)

locations. The augmented BEV features enable the detection head to access high-resolution z-axis information, resulting in the improvement of the 3D detection performance of the network.

Our contributions can be summarized as follows:

- We propose a new framework for 3D object detection where the BEV feature map is augmented with its FV counterpart, thus enriching the BEV features with high-resolution z-axis information.
- We introduce a new module termed multiview attention (MVA) to augment BEV features with FV features. The MVA module is designed to be compatible with any 3D object detection networks that use the popular 3D voxel volume to 2D BEV feature map feature extraction process. Therefore, any system that uses such networks can experience improvement in its 3D detection performance with only small updates in its code, limiting the possibility of bug introduction on already-deployed systems.
- We conduct evaluations on four popular existing 3D object detection networks augmented with our MVA modules on the widely used KITTI dataset. Evaluation results show that our module successfully improves those networks. In addition, we perform an ablation study to figure out the effects of different architectures and hyperparameters in the MVA module.

The remaining of this paper is structured as follows: we discuss relevant prior works in Section II, introduce the structure of 3D object detection with our MVA module in Section III, discuss our experimental results in Section IV, and conclude the paper in Section V.

II. Related Works

Various 3D object detection networks have been introduced in recent years. In general, the existing networks can be classified into three different categories based on their feature extraction approach, i.e., voxel-based, point-based, or hybrid approach.

A. Point-based

In the point-based approaches, the feature extractor of the network is conditioned to learn directly from a set of raw points without discretizing the point cloud as in the voxel-based approach. To achieve this objective, networks such as Point-RCNN [2] utilizes PointNet++ [3] to learn the point-wise features. One advantage of using point-based feature extractors is that they preserve high-resolution spatial structure information of the points as there is no discretization process. However, point-based approaches often suffer from expensive computations as the number of points in a point cloud is often enormous. Therefore, point-based methods are often slower than their voxel-based counterparts.

B. Voxel-based

In the voxel-based approaches, points in a point cloud are first discretized into non-overlapping cuboids of identical sizes. The feature extractor is then conditioned to learn features for each voxel according to the points that lie inside. The voxel-based approach was first popularized by VoxelNet [4], where the features of each voxel are learned by a voxel feature encoder module. Meanwhile, SECOND [16] introduced the use of sparse 3D convolution to replace the computationally expensive 3D convolution operations, leading to a significant improvement in the inference speed.

Recent voxel-based methods try to improve the training scheme or the second stage network. For example, SE-SSD [5] introduced the teacher-student learning scheme to train the networks with soft targets, while Voxel-RCNN [13] proposed a fast query technique to obtain voxel features to be used in the second stage refinement.

C. Hybrid-based

Hybrid approaches use both the point-based and voxel-based features to create rich feature vectors. PV-RCNN [14] first extract the voxel features and BEV features by utilizing the sparse 3D convolutions and 2D feature pyramid networks (FPN) [6], respectively. In the second stage, the network queries raw point features, voxel features, and BEV features of the object proposals to refine the bounding box predictions. Similarly, PVGNet [7] stacks together point-based features from the raw points, voxel-based features from the voxel feature volume, and grid features from the BEV feature map to create enriched feature vectors. BtcDet [17] on the other hand augments the raw point features and voxel features with occupancy features predicted by an auxiliary network.

III. Methodology

In this section, the problem of 3D object detection in a point cloud will first be defined. Following that, we explain the common structure of voxel-based 3D object detection networks that we used in our experiments. Last, we describe in detail how our MSA module works.

A. Problem Definition

A Lidar point cloud $\boldsymbol{P}$ is defined as a set of N^p points, $\boldsymbol{P} = \{\boldsymbol{p}_1, ..., \boldsymbol{p}_{N^p}\}$. A point $\boldsymbol{p}_i$ can be described as a feature vector $\boldsymbol{p}_i = [x_i, y_i, z_i, \boldsymbol{f}_i^{raw}] \in \mathbb{R}^{3+C^{raw}}$ with (x_i, y_i, z_i) as the 3D coordinates of the point and $\boldsymbol{f}_i^{raw}$ is an additional feature vector of C^{raw} dimension that describes the point, such as intensity, reflectivity, or ring information. An object m in a point cloud can be described by its class s_m and bounding box $\boldsymbol{b}_m$. In most dataset, $\boldsymbol{b}_m$ is constructed by using the center coordinates of the object (x_m, y_m, z_m), its dimension (d_m, w_m, h_m), and its yaw angle θ_m.

Using previously described notations, the basic objective of the 3D object detection with a deep neural network is to find the best parameters of a learnable function $\mathbf{\Phi}$ that is conditioned on the point cloud $\mathbf{P}$ to predict a set of classes $\mathbf{S}$ and bounding boxes $\mathbf{B}$ of objects that are present in the scene. The optimization objective becomes,

$$\Theta_{MLE} = argmax_{\Theta}(\mathcal{P}(\boldsymbol{S}, \boldsymbol{B}|\boldsymbol{P})), \tag{1}$$

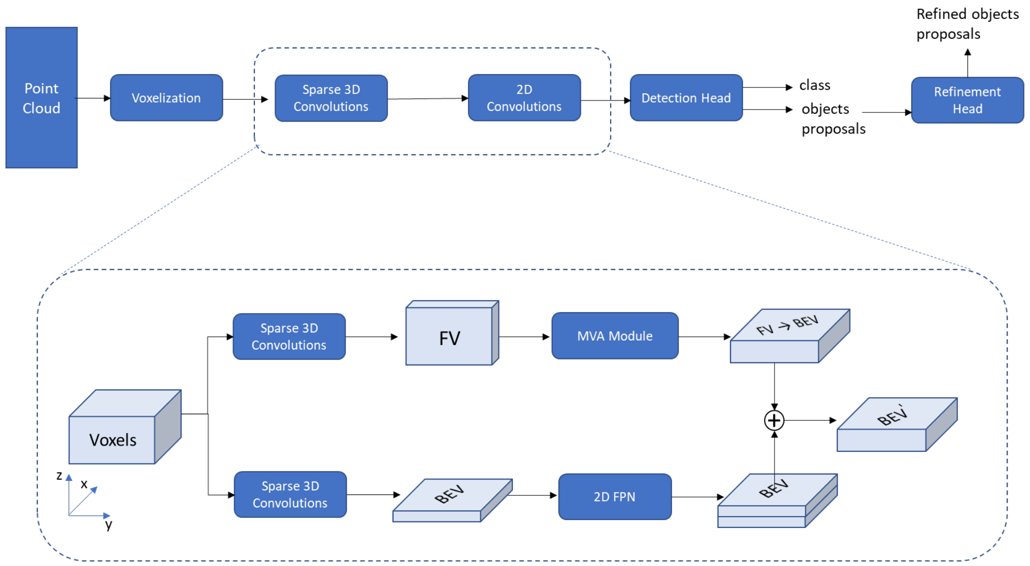


Fig. 1. An overview of the general voxel-based 3D object detection networks with our proposed MVA module. We modify the common sequence of sparse 3D convolutions and 2D convolutions into a two-branch sparse 3D convolutions, 2D convolutions (FPN), and an MVA module.

and during inference, the learnable function $\boldsymbol{\Phi}$ will produce outputs,

$$\boldsymbol{\Phi}(\boldsymbol{P}) = \{(\hat{s}_1, \hat{\boldsymbol{b}}_1), ..., (\hat{s}_{M^P}, \hat{\boldsymbol{b}}_{M^P})\}. \tag{2}$$

Note that the number of predicted objects M^p is not necessarily equal to the number of actual objects (ground truths) M that are present in the scene.

B. Voxel-based Framework for 3D Object Detection

In this work, we design our MVA module to be compatible with any 3D object detection network that uses the popular voxel-based feature extractor. Figure 1 shows the overall structure of a general 3D object detection network with the addition of our MVA module. Traditional voxel-based 3D object detection networks generally start with a voxelization module, followed by 3D sparse convolutions and 2D convolutions, and finally a detection head that produces object proposals. In the case of two-stage detection network, object proposals from the first stage are used by the refinement head to refine the bounding box proposals.

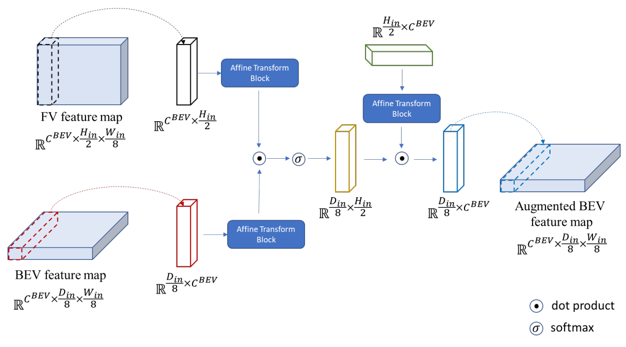
Voxelization As shown in Figure 1, a 3D object detection network that uses voxel-based framework will first discretize the point cloud $\mathbf{P}$ into non-overlapping voxels of equal sizes which will results in a set of non-empty voxels $\boldsymbol{V}^{raw} = \{\boldsymbol{v}_1^{raw}, ..., \boldsymbol{v}_{N^v}^{raw} | \boldsymbol{v}_i^{raw} \in \mathbb{R}^{K\times(3+C_{raw})}\}$, where N^v is the number of non-empty voxels and K is the predetermined number of points in a voxel. As the original number of points in a voxel, K_0, varies, we apply zero-padding to create "fake" points if $K_0 < K$ and random sampling if $K_0 > K$. All non-zero points that lie in the voxel $\boldsymbol{v}_i^{raw}$ will be averaged to create a single feature vector so that the voxels become $\boldsymbol{V}^{mean} = \{\boldsymbol{f}_1^{mean}, ..., \boldsymbol{f}_{N^v}^{mean} | \boldsymbol{f}_i^{mean} \in \mathbb{R}^{3+C^{raw}}\}$.

The non-zero averaging is used to reduce the effect of zero-padding towards feature vectors in the subsequent layers. If the voxels contain feature vectors from "fake" points, then an all-zero feature vector from a "fake" point may be transformed into a feature vector with non-zero values by any affine transformation that we apply in subsequent layers. Such non-zero feature vectors coming from non-existent points may be considered as noises to our network. Performing non-zero averaging guarantees that the mean feature vectors always come from existing points. We can maintain the representativeness of the mean feature vectors by setting the voxel size small enough such that there are only small variances in the raw feature vectors of the points inside each voxel.

Two-way Sparse 3D Backbone Convolution in the 3D space is computationally expensive, especially when the operation is performed on a large volume. To speed up the process of 3D convolution, we leverage the fact that point cloud data are often sparse and apply the widely used sparse and submanifold 3D convolutions [11] that operate only on the non-zero elements of the feature volume.

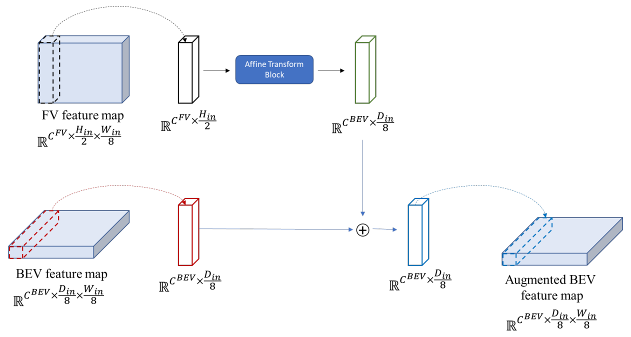
We define the spatial shape of a voxel volume $\mathbf{V}$ as the number of voxels in each axis of the 3D coordinates that are required to cover the region of interest in the point cloud. For input voxels $\boldsymbol{V}^{mean}$ with a spatial shape of $D^{in} \times W^{in} \times H^{in}$, the sparse 3D convolutional backbone will produce two 2D feature maps, FV and BEV. In our experiments, the default spatial shape of the FV is $1 \times \frac{W^{in}}{8} \times \frac{H^{in}}{2}$, while the default spatial shape of the BEV is $\frac{D^{in}}{8} \times \frac{W^{in}}{8} \times 1$. To create the two maps, we apply two branches of sparse 3D convolutional blocks, as shown in Figure 1. The upper branch is responsible for the FV feature map, while the lower branch is responsible for the BEV feature map.

The FV feature map has high-resolution in the z-axis, the BEV feature map has high-resolution in the x-axis, and both feature maps have the same dimension in the y-axis. Therefore, combining FV features to the BEV features based on their



$\mathbb{R}^{\frac{H_{in}}{2} \times C^{BEV}}$
Affine Transform Block
Affine Transform Block
FV feature map
$\mathbb{R}^{C^{BEV} \times \frac{H_{in}}{2} \times \frac{W_{in}}{8}}$
$\mathbb{R}^{C^{BEV} \times \frac{H_{in}}{2}}$
$\mathbb{R}^{\frac{D_{in}}{8} \times \frac{H_{in}}{2}}$
$\mathbb{R}^{\frac{D_{in}}{8} \times C^{BEV}}$
Augmented BEV feature map
$\mathbb{R}^{C^{BEV} \times \frac{D_{in}}{8} \times \frac{W_{in}}{8}}$
Affine Transform Block
BEV feature map
$\mathbb{R}^{C^{BEV} \times \frac{D_{in}}{8} \times \frac{W_{in}}{8}}$
$\mathbb{R}^{\frac{D_{in}}{8} \times C^{BEV}}$
dot product
softmax

Fig. 2. MVA with Dot Product Attention



Affine Transform Block
FV feature map
$\mathbb{R}^{C^{FV} \times \frac{H_{in}}{2} \times \frac{W_{in}}{8}}$
$\mathbb{R}^{C^{FV} \times \frac{H_{in}}{2}}$
$\mathbb{R}^{C^{BEV} \times \frac{D_{in}}{8}}$
BEV feature map
$\mathbb{R}^{C^{BEV} \times \frac{D_{in}}{8} \times \frac{W_{in}}{8}}$
$\mathbb{R}^{C^{BEV} \times \frac{D_{in}}{8}}$
$\mathbb{R}^{C^{BEV} \times \frac{D_{in}}{8}}$
Augmented BEV feature map
$\mathbb{R}^{C^{BEV} \times \frac{D_{in}}{8} \times \frac{W_{in}}{8}}$

Fig. 3. MVA with Affine Transformation

location on the y-axis should yield meaningful features that contain high-resolution z-axis information while still maintaining the BEV shape. The combination procedure is explained in detail in the Multiview Attention Module subsection.

2D Convolutional Backbone, Detection Head, and Refinement Head After obtaining both FV feature map and BEV feature map via two-way sparse 3D convolutions, the BEV feature map is further processed with a 2D feature pyramid network (FPN) [6] as in recent 3D object detection networks [13]–[16]. The 2D FPN enables the network to capture global structural relationships between the BEV features. The FPN output of BEV feature map is subsequently augmented with the FV feature map by leveraging our proposed MVA module. The details of this combination process are explained in the next subsection. Given the augmented BEV feature map from the MVA output, the detection head performs 1×1 convolutions on each feature vector $\boldsymbol{f}_i^{BEV}$ on the map to predict its class, s_i and bounding box proposal $\boldsymbol{b}_i$. In the case of two stage detectors, the bounding box proposals are further refined by the refinement head by leveraging features from different sources such as raw points, voxels, and BEV map.

C. Multiview Attention Module

An object in the 3D space should occupy the same y-axis coordinate in both FV and BEV. In other words, FV and BEV features that lie on the same y-axis coordinate should represent the same objects and environments. We leverage this fact to augment the BEV features with the FV features. Given an FV feature map $\boldsymbol{f}^{FV}$ of size $H^{out} \times W^{out} \times C^{out}$ and a BEV feature map $\boldsymbol{f}^{BEV}$ of size $D^{out} \times W^{out} \times C^{out}$, the MSA module takes the features residing on the i-th index of the y-axis, $\boldsymbol{f}_i^{FV} \in \mathbb{R}^{H^{out} \times C^{out}}$ and $\boldsymbol{f}_i^{BEV} \in \mathbb{R}^{D^{out} \times C^{out}}$, and combine the two features together to make the augmented BEV features $\boldsymbol{f}_i^{BEV'} \in \mathbb{R}^{D^{out} \times C^{out}}$. We provide two ways of combining the features, via dot-product attention mechanism or via a single affine transform block applied to the FV features.

MVA with Dot Product Attention The process of constructing the augmented BEV features $\boldsymbol{f}_i^{BEV'}$, shown in Figure 2, can be described as transforming a sequence of feature vectors $\boldsymbol{f}_i^{FV}$ of length H^{out} into a new sequence $\boldsymbol{f}_i^{FV'}$ of length D^{out}. The transformed sequence is combined with another sequence $\boldsymbol{f}_i^{BEV}$ of length D^{out} via an aggregating function such as a summation. For the dot-product attention, we design the dimension size of $\boldsymbol{f}_i^{FV}$ channels to be the same as $\boldsymbol{f}_i^{BEV}$ channels, C^{BEV}.

The dot product attention mechanism has been widely used for sequence-to-sequence transformation. In recent years, the multihead variant of the dot product attention mechanism was popularized by [8] in the natural language processing domain and [9] in the computer vision domain. In this work, we utilize the multihead attention mechanism, defined as,

$$MHA(Q, K, V) = Concat(h_1, ..., h_{N^h})W^{out}, \tag{3}$$

where,

$$h_k = Att(\boldsymbol{f}_i^{BEV} W_k^Q, \boldsymbol{f}_i^{FV} W_k^K, \boldsymbol{f}_i^{FV} W_k^V), \tag{4}$$

$$Att(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V, \tag{5}$$

$d_k = C^{BEV}/N^h$, and Ws are learnable weight matrices.

Intuitively, the attention mechanism figures out the importance of each element in $\boldsymbol{f}_i^{FV}$ with regard to $\boldsymbol{f}_i^{BEV}$ in the form of a weight matrix via softmax and scaled-correlation of the two. The weight matrix is then used to construct a new sequence $\boldsymbol{f}_i^{FV'}$ from $\boldsymbol{f}_i^{FV}$ values but in the shape of $\boldsymbol{f}_i^{BEV}$. Combining the sequence of feature vectors $\boldsymbol{f}_i^{FV'}$ with $\boldsymbol{f}_i^{BEV}$ via an aggregation function, in our case a summation, will result in an augmented BEV feature vectors $\boldsymbol{f}_i^{BEV'}$. The augmented BEV feature map provides a way for the detection head to access high-resolution z-axis information that is not available with only a BEV feature map.

MVA with Affine Transform The dot-product attention mechanism requires at least three different affine transform blocks per head and requires large memory space as observed in many transformer-based architectures. We provide an alternative solution where we only use a single affine transform block without dot product operation between the sequence of feature vectors, as shown in Figure 3. In this case, $\boldsymbol{f}_i^{FV}$ is directly transformed into $\boldsymbol{f}_i^{FV'}$, which has the same shape as $\boldsymbol{f}_i^{BEV}$, by a single layer of affine transformation. This mechanism can be seen as a location-based attention introduced in [18]. This alternative is more efficient compared

TABLE I
EVALUATION RESULTS OF VARIOUS 3D OBJECT DETECTION NETWORKS WITH AND WITHOUT THE PROPOSED MVA MODULE ON THE KITTI *val* SET. THE VALUES ARE FOR %AP WITH 11 SAMPLING POINTS. FOR NETWORKS WITH MVA, THE SECOND ROW INDICATES THE PERFORMANCE DIFFERENCE COMPARED TO THE ORIGINAL NETWORK.

	Car			Pedestrian			Cyclist			Mean
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard	
Voxel-RCNN [13]	89.21	83.41	78.60	67.19	60.65	55.16	85.73	72.21	68.46	73.40
Voxel-RCNN with MVA	89.77	84.06	78.99	67.02	61.25	55.41	86.28	72.42	68.51	73.74
	+0.56	+0.65	+0.39	-0.17	+0.60	+0.25	+0.55	+0.21	+0.05	+0.34
PV-RCNN [14]	89.33	83.61	78.71	63.10	54.82	51.77	86.06	69.48	64.54	71.27
PV-RCNN with MVA	89.17	84.58	78.66	65.44	57.97	53.80	86.36	72.68	69.25	73.10
	-0.16	+0.97	-0.05	+2.34	+3.15	+2.03	+0.30	+3.20	+4.71	+1.83
PV-RCNN++ [15]	88.88	79.04	78.26	64.00	59.40	54.49	86.76	66.94	65.69	71.50
PV-RCNN++ with MVA	89.06	83.56	78.35	65.14	61.60	55.96	86.84	67.31	65.50	72.59
	+0.18	+4.52	+0.09	+1.14	+2.20	+1.47	+0.08	+0.37	-0.19	+1.09
SECOND [16]	88.61	78.62	77.21	56.54	52.98	47.73	80.58	67.13	63.10	68.06
SECOND with MVA	88.93	79.11	77.88	61.75	55.92	49.99	85.57	70.44	64.86	70.49
	+0.32	+0.49	+0.67	5.21	+2.94	+2.26	+4.99	+3.31	+1.76	+2.43

TABLE II
MEAN OF THE %AP DIFFERENCE BETWEEN NETWORKS WITH MVA AND THEIR ORIGINAL NETWORK

Car			Pedestrian			Cyclist		
Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
+0.23	+1.65	+0.27	+2.13	+2.22	+1.50	+1.48	+1.77	+1.58
+0.72			+1.95			+1.61		

to the dot-product attention mechanism, however it has lower model capacity. We show the effects between utilizing dot-product attention and affine transform in the ablation study.

IV. EXPERIMENTAL SETUPS AND RESULTS

In this section, we will describe the dataset that we used to conduct our experiments and list all implementation details that are needed to reproduce the experimental results. We then show and discuss our evaluation results on the validation set.

A. Dataset

In this work, we utilize the widely used KITTI dataset [1] for 3D object detection with Lidar point cloud. The KITTI dataset is one of the earliest and most popular datasets for 3D object detection in Lidar point cloud. The dataset contains 7,481 Lidar frames for training and 7,518 Lidar frames for testing. There are three major classes in the dataset: Car, Pedestrian, and Cyclist. For every object sample of each class, we can assign a difficulty level, i.e. easy, moderate, or hard, depending on the level of occlusions of said object.

As the annotation for test set is not publicly available, we split the training data into *train* split with 3,712 frames and *val* split with 3,769 frames. We follow the commonly used protocol for splitting the KITTI training data [10] such that the frames in the *train* split and *val* split originate from different sequences. We perform extensive evaluations on the *val* set, not the *test* set, as per KITTI's official rules for works that are a modification of existing techniques.

B. Implementation Details

We set the number of epochs as 80, batch size as 4, and use Adam [12] as the optimizer. The one cycle scheduler is used to control the learning rate, where the maximum learning rate, 0.01, is set to be achieved at about halfway through the training. For all networks, we define the voxel size as $5cm \times 5cm \times 10cm$ for the x, y, and z axis, respectively, and limit the detection range to be $0 \sim 70m$ for the x-axis, $-40 \sim 40m$ for the y-axis, and $-3 \sim 1m$ for the z-axis relative to the position of the Lidar sensor on the ego car.

To make fair comparisons between networks with MVA and their original counterparts, we retrain the original networks using the publicly available code with the same hyperparameters as the ones with MVA modification. We also refer to the original publication of each network for the loss functions.

C. Main Results

Table I shows the evaluation results of various voxel-based 3D object detection networks with and without the proposed MVA module on the *val* set of the KITTI dataset. We choose to use the dot-product MVA module with 8 heads following the ablation results, which is explained in details in the next subsection. As shown on the table, our MVA module improves the 3D detection performance of voxel-based 3D object detection networks in general. More specifically, our module improve the detection performance of all four networks by 0.72% for car class, 1.95% for pedestrian class, and 1.61% for cyclist class, as shown in Table II.

It is interesting to note that the improvements in both pedestrian and cyclist classes are higher compared to the improvement in the car class. This phenomena can be explained by the bounding box dimensions of the objects on the road. Cars in general have similar shapes and sizes as car manufacturers have to follow existing regulations, therefore, their bounding box dimensions are similar for all samples in the dataset. On

TABLE III
COMPARISON BETWEEN DOT-PRODUCT AND AFFINE TRANSFORMATION MVA MODULES WITH VOXEL-RCNN-MVA

	Car			Pedestrian			Cyclist			Overall Inference
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard	Time (ms)
Dot Product	**89.77**	**84.06**	**78.99**	**67.02**	**61.25**	**55.41**	**86.28**	**72.42**	**68.51**	33.6
Affine Transform	89.60	83.97	78.88	65.56	60.05	54.97	84.95	72.25	68.22	**30.4**

TABLE IV
EFFECTS OF THE NUMBER OF HEADS IN THE DOT-PRODUCT MVA MODULE WITH VOXEL-RCNN-MVA

Number of	Car			Pedestrian			Cyclist			Overall Inference
Heads	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard	Time (ms)
4	89.65	83.96	78.91	65.05	58.70	52.89	85.90	72.38	68.48	**33.1**
8	**89.77**	**84.06**	**78.99**	**67.02**	**61.25**	**55.41**	86.28	72.42	68.51	33.6
16	89.37	83.72	78.75	66.14	60.06	55.22	**92.07**	**73.25**	**69.30**	33.8

the contrary, pedestrians and cyclists have varying bounding box dimensions, particularly in the z-axis due to the variance in people's heights. As such, the importance of obtaining high-resolution information in the z-axis from the FV features is far greater for predicting bounding boxes of pedestrians and cyclists compared to bounding boxes of cars.

D. Ablation Study

We conduct ablations on the Voxel-RCNN network for two aspects: the differences between the two MVA modules and the number of heads in the dot-product MVA module. As shown in Table III, the dot-product MVA module performs better compared to the affine transformation MVA module in terms of %AP. However, this method has a slower overall inference time of about 3.2ms (10.5%). This phenomena is expected, and has been previously explained in Subsection 3.C.

Another hyperparameters that can affect networks performance is the number of heads in the dot-product MVA module. Table IV shows the effects of number of heads on the network performance. As expected, the processing time gets slower when the number of heads is increased. However, higher number of heads does not necessarily means the network has a better performance in terms of %AP. We find that the network performs optimally when we set the number of heads as 8.

Note that the original Voxel-RCNN has an overall inference time of 29.6 ms, meaning that utilizing the two-way 3D sparse convolutions and either the affine transform MVA or dot-product MVA only adds 0.8 ms (2.7%) or 4 ms (13.5%) to the overall inference time, respectively.

V. CONCLUSIONS

We present the Multiview Attention Module (MVA) for 3D object detection that augments a bird-eye-view (BEV) feature map with features from a front-view (FV) feature map. The feature augmentation process enable the detection head to obtain high-resolution information in the z-axis while still allowing object predictions to be made on the BEV. As such, we successfully maintain reasonable inference speed while simultaneously improve the 3D detection performance. Based on the experimental results on the KITTI *val* set, our MVA module successfully improve the detection performance of all four voxel-based networks that we evaluated, proving the effectiveness and the adaptability of the MVA module.

REFERENCES

[1] A. Geiger, P. Lenz, R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," 2012 IEEE conference on computer vision and pattern recognition, 2012, pp. 3354-3361

[2] S. Shi, X. Wang, H. Li, "Pointrcnn: 3d object proposal generation and detection from point cloud," Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 770-779

[3] C.R. Qi, L. Yi, H. Su, L.J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," Advances in Neural Information Processing Systems 30, 2017.

[4] Y. Zhou, O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 4490-4499

[5] W. Zheng, W. Tang, L. Jiang, C.W. Fu, "SE-SSD: Self-Ensembling Single-Stage Object Detector From Point Cloud," Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 14494-14503

[6] T.Y. Lin, et al., "Feature pyramid networks for object detection," Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 2117-2125

[7] Z. Miao, et al., "PVGNet: A Bottom-Up One-Stage 3D Object Detector With Integrated Multi-Level Features," Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 3279-3288

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, "Attention is all you need," Advances in neural information processing systems, 2017, pp. 5998-6008

[9] A. Dosovitskiy, et al, "An image is worth 16x16 words: Transformers for image recognition at scale," arXiv preprint arXiv:2010.11929, 2020

[10] X. Chen, et al., "3d object proposals for accurate object class detection," Advances in Neural Information Processing Systems, 2015, pp. 424-432

[11] B. Graham, L. van der Maaten, "Submanifold sparse convolutional networks," arXiv preprint arXiv:1706.01307, 2017

[12] D. P. Kingma, J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014

[13] J. Deng, et al., "Voxel R-CNN: Towards High Performance Voxel-based 3D Object Detection," arXiv preprint arXiv:2012.15712 (2020)

[14] S. Shi et al., "PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection," 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 10526-10535

[15] S. Shi, et al., "PV-RCNN++: Point-Voxel Feature Set Abstraction With Local Vector Representation for 3D Object Detection," arXiv preprint arXiv:2102.00463, 2021

[16] Y. Yan, Y. Mao, B. Li, "Second: Sparsely embedded convolutional detection," Sensors 18, no. 10, 2018

[17] Q. Xu, Y. Zhong, U. Neumann, "Behind the Curtain: Learning Occluded Shapes for 3D Object Detection," arXiv preprint arXiv:2112.02205, 2021

[18] M. T. Luong, H. Pham, C. D. Manning, "Effective approaches to attention-based neural machine translation," arXiv preprint arXiv:1508.04025, 2015