Received 3 September 2022, accepted 21 September 2022, date of publication 26 September 2022, date of current version 6 October 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3210108

RESEARCH ARTICLE

MD3D: Mixture-Density-Based 3D Object Detection in Point Clouds

JAESEOK CHOI[1], YEJI SONG[1], YERIM KIM[1], JAEYOUNG YOO[2], AND NOJUN KWAK[1], (Senior Member, IEEE)
[1]Department of Intelligence and Information, Seoul National University, Seoul 08826, South Korea
[2]NAVER WEBTOON AI, Seongnam 15329, South Korea

Corresponding author: Nojun Kwak (nojunk@snu.ac.kr)

This work was supported by the Government of the Republic of Korea under NRF Grant 2021R1A2C3006659 and IITP Grant 2022-0-00953.

ABSTRACT The design factors of anchor boxes, such as shape, placement, and target assignment policy, greatly influence the performance and latency of the 3D object detectors. Unlike image-based 2D anchors, 3D anchors must be placed in a 3D space and determined differently for each class of different sizes. This imposes a significant burden on the design complexity. To tackle this issue, various studies have been conducted on how to set the anchor form. However, for practical reasons, anchor-based methods select the anchor design by compromising between performance and latency. Consequently, only objects that are similar in shape and size to an anchor can obtain high accuracy. In this paper, we propose a Mixture-Density-based 3D Object Detection (MD3D) in point clouds to predict the distribution of 3D bounding boxes using a Gaussian Mixture Model (GMM). With an anchor-free detection head, MD3D requires few hand-crafted design factors and eliminates the inefficiency of separating the regression channel for each class, and thus offering both latency and memory benefits. MD3D is designed to utilize various types of feature encoding; therfore, it can be applied flexibly by replacing only the detection head of the existing detectors. Experimental results on the KITTI and Waymo open datasets show that the proposed method outperforms its counterparts that are based on the conventional anchor-based detection head in its overall performance, latency, and memory. The code is publicly available at `https://github.com/sky77764/MD3D`

INDEX TERMS Autonomous vehicles, object detection, visual perception.

I. INTRODUCTION

Recently, the industrial demand for autonomous vehicles and robotics technology is increasing rapidly. With this rising demand, various sensors, such as monocular cameras, stereo cameras, light detection and ranging(LiDAR), and solid-state LiDAR, have been developed to capture the world's 3D spatial information in data. LiDAR enables more accurate and sophisticated distance measurements than other sensors, hence it has been widely used for the development of 3D object detection methods.

Raw point clouds obtained by LiDAR sensors have noisy sparse representation with an imbalance sampling problem, which causes many occluded surfaces to be without any points. Therefore, methods have been devised to compensate for the weaknesses of point clouds at various levels. Several methods have been proposed to enhance the performance of object detectors at the input level [1], [2]. At the feature level, there are various representation forms, such as raw points [3], [4], [5], [6], voxels [7], [8], [9], [10], graphs [11], and their hybrids [12], [13], [14]. Two streams of works based on the presence or absence of anchor boxes have been proposed at a higher level. The former is known as an *anchor-based* method [7], [10], [12], [13], [15], [16], and the latter is an *anchor-free* method [5], [17], [18], [19], [20], [21].

The existing 3D object detectors have mostly adopted anchor-based detection methods. In this study, we present the anchor or anchor box as a set of predefined 3D boxes for each object class by using the scale and aspect ratio of the class. Anchors are tiled across the scene (gray boxes in

The associate editor coordinating the review of this manuscript and approving it for publication was J. Jun Cheng.

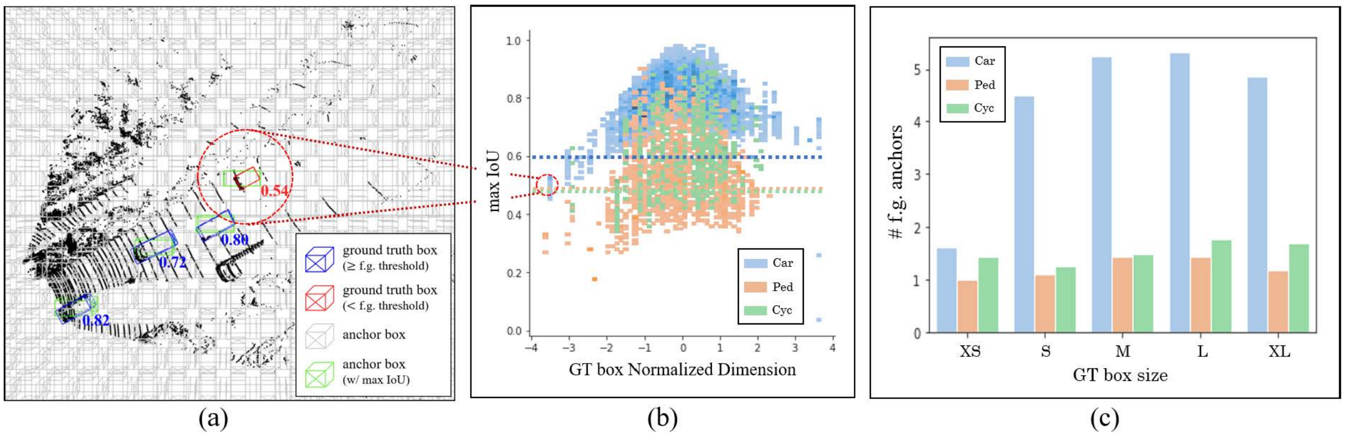


FIGURE 1. Problems caused by hand-crafting factors of anchor boxes. (a) shows the BEV of a point cloud with the equally spaced anchors and ground truth (GT) boxes of the car class. The values under each GT represent the maximum IoU with anchors. Most cars have average sizes that match the anchor well, so the IoU value exceeds the foreground threshold (blue). However, some cars far from the average may not exceed the threshold (red). (b) shows the IoU value with the best-matched anchor box for all GT boxes in KITTI-*train* data. The farther the size of the GT box is from the mean size (0), the lower the maximum IoU of the box. Dotted lines represent foreground IoU thresholds for each class. If the maximum IoU of a GT box does not exceed the threshold, at least one anchor box with the highest IoU value is forced to be assigned for each GT. (c) the average number of assigned anchors for each GT box size group. It shows how the number of anchors assigned to foregrounds varies by class and size.

Fig. 1 (a)), and anchor-based detectors detect target objects by predicting offsets from nearby anchors for each class. However, anchor-based detectors suffer from a fatal problem; they must predefine many anchor-related hyper-parameters: 1) anchor size, 2) anchor direction, 3) stride that determines anchor placement, and 4) target assignment policy. Each of these factors dominantly influences the performance and latency of the detectors. Thus, they must be defined separately for each class.

Existing anchor-based methods usually set their default anchor size as the average of all ground truth (GT) bounding boxes with 0 and 90 degrees rotations. Unifying the anchor size as a representative value, *i.e.,* the average size of objects in the training data, would be a good compromise in the trade-off between speed and performance. However, objects with significant deviations from the default anchor size inevitably tend to be overlooked. Another hyper-parameter that biases a detector is the stride between neighboring anchors. The spatial dimensions of the last feature map determine the anchor stride. For example, in the KITTI dataset [22], a typical compact detector has a 200×176 feature map; the anchor-to-anchor interval is about 0.4m, which is relatively narrow for large objects such as *Car*. Contrastingly, for small objects, such as *Pedestrian* and *Cyclist*, it is too wide to cover all GTs. Accordingly, the IoU value between an anchor and a GT varies significantly for each object class, and hence the foreground threshold is inevitably set differently for each class. In addition, calculating the IoU in a 3D space can result in extremely low IoU values, leading to multiple placements of anchors on the z-axis (vertical direction). To mitigate this problem, conventional methods ignore the z-axis using Bird's Eye View (BEV) 2D IoU.

Fig. 1 shows the problems caused by hand-crafting factors in the anchor-based detection methods in detail. Notably, the average number of assigned anchors for Car with an extreme size (XS in (c)) is much smaller than the average-sized Car. Consequently, outlier objects with extreme sizes show low recognition rates compared to objects within the normal range because of an insufficient number of assigned anchors. Moreover, regardless of the GT box size, the number of foreground anchors for Car is overwhelmingly larger than those for the other two classes. This is because the strides of the anchors cannot be assigned differently for each class. A relatively larger stride for Pedestrian and Cyclist is likely to cause the objects belonging to these classes to be unable to match with any of the anchor boxes. Therefore, an unsuitable anchor box with a low IoU value, which is also the highest among the other anchor boxes, is forcibly assigned to avoid the zero assignment. Through this, it can be pointed out that it is not the number of GT boxes or other biases in the training data but the inherent structural limitations of the anchor-based detection methods that severely affect the inferior detection performance of Pedestrian and Cyclist.

Our proposed Mixture Density network for 3D Object Detector (MD3D) is a method of estimating the distribution of 3D bounding boxes in point clouds with a Gaussian Mixture Model (GMM), which is free from the problems experienced by anchor-based detectors mentioned above. Another significant merit of the MD3D is that it is free from the discrepancy between classification and regression loss. Most 3D object detectors use the focal loss [23] for a classification loss to adjust their weights according to the estimation accuracy, allowing them to learn well about data with fewer samples. By contrast, regression loss treats the anchors assigned as the foreground equally. Therefore, with typical regression loss, it is inevitable that classes whose GT box shapes are concentrated close to the mean, *i.e.,* Cars in the KITTI dataset, have superior performance. Our MD3D estimates the 3D bounding box in a distribution form throughout the scenes without any process of assigning the GT during regression learning and without distinguishing classes or box sizes. Thus, it can reduce heuristic design factors and cover a wider variety of data samples. The contributions of this study are as follows.

- Among point-cloud-based 3D object detectors, we first propose an anchor-free detection method that estimates the density of bounding boxes and no longer requires a heuristic ground truth assignment.
- Our proposed MD3D is applicable to any type of point cloud feature encoding methods that enables it to be plugged and played easily to the existing detectors.
- MD3D shows superior performance and latency compared to the existing detection heads and facilitates learning by minimizing hand-crafted design factors.

II. RELATED WORK

A. 3D OBJECT DETECTION IN POINT CLOUDS

As point clouds provide accurate geometric scene information, point cloud-based methods have achieved high performance in 3D object detection. However, the inherent properties of irregular and sparse point clouds impose difficulties in data processing. Therefore, various forms of point cloud representations have been proposed. SECOND [7] groups the point clouds into voxels and utilizes spatially sparse convolution, thereby reducing the computational burden of the 3D convolution. PointPillars [15] encodes point clouds into stacked pillars and operated all processes with only 2D convolutions, removing the bottleneck of 3D convolutions. PointRCNN [5] directly learns representations from raw point clouds using PointNet++ [3], [4] as a backbone network and generates bounding box proposals efficiently by taking advantage of point segmentation, which provided a powerful clue to 3D object detection. VoteNet [24] also uses PointNet++ as a backbone and detects objects using a deep Hough voting method. PV-RCNN [12] utilizes both voxel-based and point-based operations to encode multi-scale features and provide accurate location information efficiently. A graph method is adpoted in [25] to detect objects.

The latest point-cloud-based methods compensate for the limitations of the existing detectors and significantly improve the accuracy and latency. Focal sparse convolution [26] predicts the importance of features in performing sparse convolution and selectively computes high importance features. IA-SSD [27] reduces the computational overhead of raw point-based detectors by using a learnable downsampling strategy. SST [28] improves the detection accuracy by introducing a single-stride backbone network that utilizes transformer blocks. Although the latest works have improved many aspects of the existing detectors, most have focused on improving the backbone structure.

Most anchor-free 3D object detectors [17], [18], [19], [20], [21] use a classification method based on heatmap estimation, which is primarily adopted in 2D object detection [29], [30], [31]; hence, they can only be used for 2D-projected features. In addition, the heatmap-based heads still have many hand-crafted design factors, such as, the Gaussian radius of the heatmap and the foreground assignment policy for regression. Because the proposed method does not require a GT assignment policy for regression, the design factors can be significantly reduced. There is no restriction on the input features, so the proposed method can be flexibly applied to various types of point cloud features.

B. MIXTURE DENSITY NETWORKS IN COMPUTER VISION

Originally, MDN [32] was proposed to predict a continuous quantity under uncertainty. The MDN has recently attracted considerable attention, especially in object detection tasks because capturing uncertainties and coping with mislocalization have become critical issues. He *et al.* [33] measured the uncertainties of bounding boxes to deal with challenging cases, such as occlusion, while Feng *et al.* [34] extended it to LiDAR 3D vehicle detection. The MDN was also utilized to model the multi-modal nature of object detection and human pose estimation [35], and to address active learning for object detection [36].

We address the problem of complex anchor design that restricts both performance and latency in 3D object detection and apply MDN to overcome this limitation. This study is inspired by MDOD [37], which reformulated the 2D object detection task as a density estimation problem, and it reduced the complex processing and heuristics in the training. We extend their works to the 3D object detection task and improve the existing detectors in a plug-and-play manner with a flexible detection head that is compatible with any representation of point clouds. Unlike images, point clouds have various feature forms (BEV, FV, voxel, point, *etc.*). The proposed MD3D can be flexibly applied to these different types of features and easily replace the detection heads of existing detectors.

III. METHOD

A. MODELING POINT-CLOUD-BASED 3D OBJECT DETECTION WITH MIXTURE DENSITY NETWORK

In point cloud-based 3D object detection, the input point cloud can be expressed as $L \in \mathbb{R}^{N\times 4}$ (3D coordinates and reflectance), and the position, size, and direction of an object can be expressed as a 3D bounding box $B \in \mathbb{R}^{N_{gt}\times 7}$. Here, N represents the number of points in the scene, and N_{gt} is the number of GT boxes. To regress object B from input L, we estimate the conditional probability distribution $p(B|L)$.

A mixture density network (MDN) [32] is a neural network, whose target is to learn the probability density function (pdf). We applied MDN to point cloud-based 3D object detection to predict the distribution of multiple bounding boxes for a given scene (point cloud), and estimate the target 3D bounding box B for the input point cloud L as a mixture model. We use the conventional GMM as the target pdf,

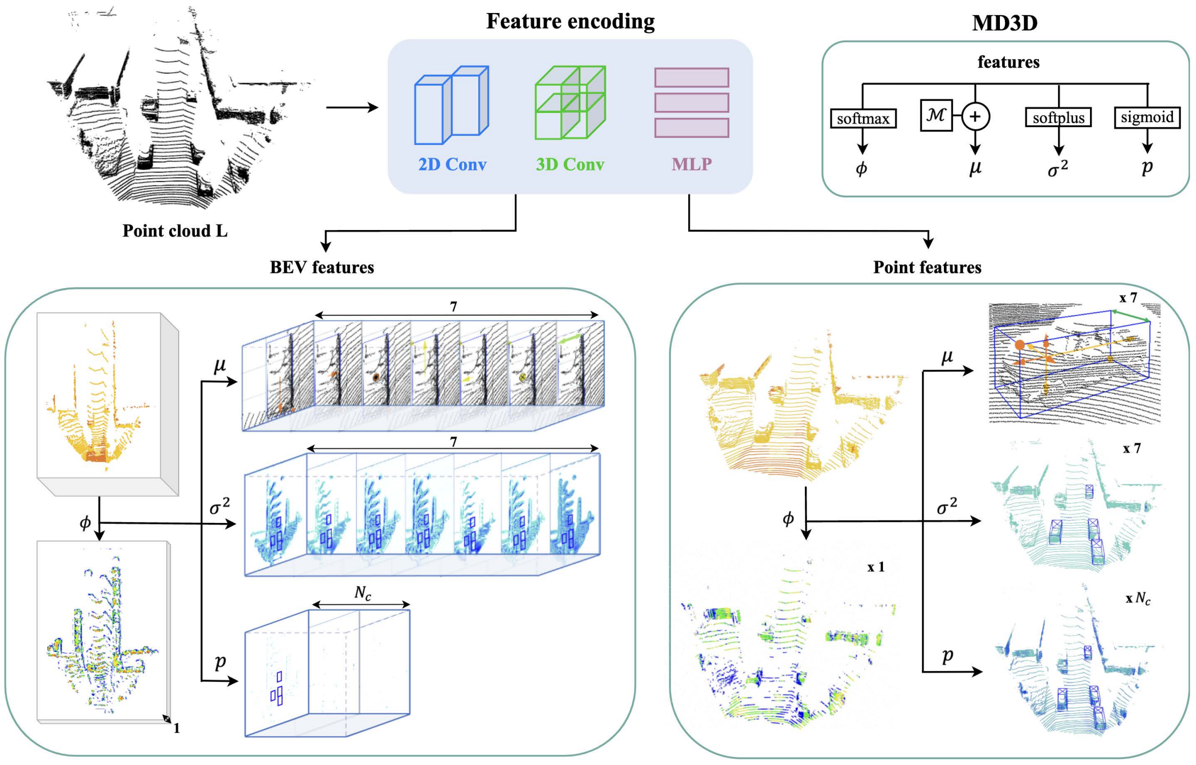


FIGURE 2. **The overall architecture of MD3D. Regardless of the encoding types and feature forms, MD3D is applicable to existing detectors in a plug-and-play manner. MD3D predicts the mixture parameters ϕ, μ, and σ^2 from a regression branch which compose the distribution of multiple bounding boxes for a given point cloud. A classification branch predicts class probability p for each class $c \in [N_c]$.**

which can be expressed as:

$$p(B|L) = \sum_{k=1}^{K} \phi_k \times \mathcal{N}(B|\mu_k, \Sigma_k) \tag{1}$$

$$\mathcal{N}(B|\mu_k, \Sigma_k) = \frac{\exp\left(-\frac{1}{2}(B-\mu_k)^\top \Sigma_k^{-1}(B-\mu_k)\right)}{\sqrt{(2\pi)^7|\Sigma_k|}} \tag{2}$$

where K is the number of mixture components, which is determined by the spatial resolution of the BEV feature or the number of point features N, and ϕ_k is the mixing coefficient. For the efficiency of the model, we assume that each element of $\mu_k \in \mathbb{R}^7$ is independent and the covariance matrix is diagonal, that is, $\Sigma_k = \text{diag}(\sigma_k^2)$ where $\sigma_k^2 \in \mathbb{R}^7$, rather than dealing with a full covariance matrix $\Sigma_k \in \mathbb{R}^{7\times7}$.

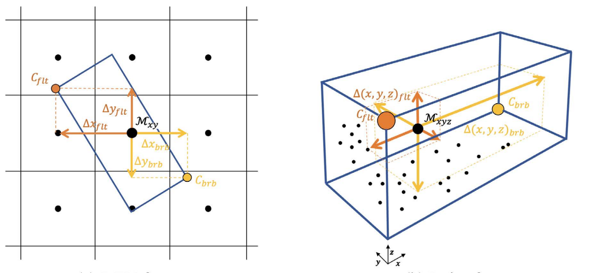
B is composed of the center position, box dimension, and yaw angle, so $B_{origin} = \{x_c, y_c, z_c, l, w, h, \theta\}$. We encode the B_{origin} as $B_{corner} = \{\boldsymbol{C}_{flt}, \boldsymbol{C}_{brb}, w\} \in \mathbb{R}^7$, which consists of the front-left-top corner $\boldsymbol{C}_{flt} = \{x, y, z\}_{flt}$, the back-right-bottom corner $\boldsymbol{C}_{brb} = \{x, y, z\}_{brb}$, and width w. Among the various ways to encode bounding box B, encoding it with two opposite corners and width can result in a more accurate regression for the bounding box. This can be attributed to the nature of the point clouds obtained by LiDAR, in which the points are not in the center of an object but are concentrated in one corner. The corner on the hindside without points can be easily regressed using peripheral point features owing to the symmetry of the target object. As part of the post-processing, we decode the B_{corner} back to the B_{origin}.

Existing anchor-based regression methods learn several B's separately in L, where each anchor's design and matching algorithm become critical elements in training. However, because our method learns by representing the distribution of multiple B's as one mixture model with the conditional distribution $p(B|L)$, unnecessary heuristic design can be eliminated.

B. NETWORK ARCHITECTURE

The MD3D consists of a regression branch that predicts three mixture parameters ϕ_k, μ_k, and σ_k^2 for $k \in [K]$, and a classification branch that predicts class probability p. The backbone of the existing 3D object detectors, which encodes the feature of a point cloud, remains unchanged. We apply the MD3D to most commonly used forms of head features, BEV-type features, and point-type features. Their structures are shown in Fig. 2; MD3D can be applied to any form and is compatible with many different methods of encoding point cloud features.

The MD3D for BEV features has a structure similar to that of MDOD [37], an MDN-based 2D object detector. The BEV feature has the shape of $H \times W \times C$, where H, W, and C represents the height, width, and number of channels, respectively. Accordingly, the number of mixture components K becomes



(a) BEV feature (b) Point feature

FIGURE 3. Illustration of corner regression. MD3D does not predict the corners (C_{flt} and C_{brb}) of bounding boxes directly but rather predicts the offsets ($\Delta(x,y,z)_{flt}$ and $\Delta(x,y,z)_{brb}$) from $\mathcal{M}_{xy}$ or $\mathcal{M}_{xyz}$, the center coordinates of the BEV feature or point feature, respectively.

$H \times W$, and the mixing coefficient $\phi \in \mathbb{R}^{H\times W\times 1}$ is forced to satisfy $\sum_k \phi_k = 1$, using softmax for the feature output. As shown in Fig. 3 (a), $\mu \in \mathbb{R}^{H\times W\times 7}$ does not predict B_{corner} directly but predicts the offsets from the center coordinates of each feature $\mathcal{M}_{xy}$. For z and w, we use the raw output rather than the offset. The process is formulated as follows:

$$\mu_{BEV} = (\mathcal{M}_x + \Delta x_{flt}, \mathcal{M}_y + \Delta y_{flt}, z_{flt}, \\ \times \mathcal{M}_x + \Delta x_{brb}, \mathcal{M}_y + \Delta y_{brb}, z_{brb}, w). \quad (3)$$

$\sigma^2 \in \mathbb{R}^{H\times W\times 7}$ predicts values greater than zero using softplus activation. $p \in \mathbb{R}^{H\times W\times N_c}$ predicts the classification probability for each class using sigmoid activation, where N_c denotes the number of classes which is set to 3 (Car, Pedestrian, Cyclist) in our experiments.

Existing anchor-based regression methods use anchors defined differently per class, and generally they use anchors in the two directions of 0 and 90 degrees. Therefore, the number of output boxes is $H \times W \times N_c \times 2$, which is $N_c \times 2$ times higher than that of our MD3D. Consequently, MD3D has advantages in terms of the number of parameters, inference time, and post-processing time.

The MD3D for the point feature has some minor modifications from that of the BEV feature because the input shape is slightly different. Because the point feature has the form of $N \times C$, where N is the number of points in a scene, the number of mixture components becomes $K = N$. Accordingly, it becomes $\phi \in \mathbb{R}^{N\times 1}$, $\mu \in \mathbb{R}^{N\times 7}$, $\sigma^2 \in \mathbb{R}^{N\times 7}$, and $p \in \mathbb{R}^{N\times N_c}$. Furthermore, as shown in Fig. 3 (b), the reference point $\mathcal{M}_{xyz}$ becomes the original (x, y, z) coordinates of the point, and μ predicts an offset from $\mathcal{M}_{xyz}$, except for w:

$$\mu_{point} = (\mathcal{M}_x + \Delta x_{flt}, \mathcal{M}_y + \Delta y_{flt}, \mathcal{M}_z + \Delta z_{flt}, \\ \times \mathcal{M}_x + \Delta x_{brb}, \mathcal{M}_y + \Delta y_{brb}, \mathcal{M}_z + \Delta z_{brb}, w). \quad (4)$$

At inference time, because the values of μ are highly likely to be close to the local maximum of the predicted GMM, we use μ of each mixture component as an independent output box. To improve the inference speed, σ^2 is not used and ϕ is used to filter out unnecessary boxes. In addition, the mixing coefficient ϕ is very low for the location where no object exists, as in the example in the Fig. 2, hence many output boxes can be filtered out. Then, using non-maximum suppression (NMS), boxes in which p is the local maximum are finally extracted.

C. LOSS FUNCTION

L_{MDN}, the regression loss, is used to learn the GMM parameters ϕ, μ, and σ^2 with a negative log-likelihood as follows:

$$L_{MDN} = -\frac{1}{N_{gt}} \sum_{n=1}^{N_{gt}} \log \left(\sum_{k=1}^{K} \phi_k \times \mathcal{N}(B_n|\mu_k, \Sigma_k) \right). \quad (5)$$

Here, N_{gt} is the number of GT bounding boxes in the scene.

For classification loss, we use the most commonly used focal loss [23], as shown below:

$$L_{focal} = -\alpha_t(1-p_t)^{\gamma} log(p_t) \\ \text{where} \quad p_t = \begin{cases} p & \text{for foreground box} \\ 1-p & \text{otherwise.} \end{cases} \quad (6)$$

Among the boxes predicted in the regression branch, when the 3D IoU of a box exceeds 0.5, we assign it to the foreground; otherwise, we assign it to the background. We use $\alpha_t = 0.25$ and $\gamma = 2$.

The loss of the MD3D head is the sum of the MDN loss and focal loss, as follows:

$$L_{MD3D} = L_{MDN} + \beta \cdot L_{focal}. \quad (7)$$

For one-stage detectors, we use MD3D loss as a final loss, and for two-stage detectors, we replace the region proposal network (RPN) loss with MD3D loss because the MD3D head is utilized in the RPN. We use $\beta = 500$.

IV. EXPERIMENT

A. DATASETS

We evaluated the proposed method on the KITTI dataset [22], one of the most popular datasets for 3D object detection for autonomous driving. It consists of 7,481 training samples and 7,518 testing samples, where the training samples are generally divided into *train* split with 3,712 samples and *val* split with 3,769 samples. Because the KITTI dataset contains only 90-degree annotation, we clipped the scenes into (0, 70.4)m, (−40, 40)m, and (−3, 1)m for the X, Y, and Z axis ranges. We also experimented on a large-scale Waymo Open dataset [38] to verify whether the performance of the MD3D improved regardless of the data size. The Waymo dataset includes 798 training sequences with approximately 160k samples and 202 validation sequences with 40k samples. Because of limited resources, we trained the models with 20% samples at regular intervals for each sequence, using a total of 32k training samples. The Waymo dataset contains a complete 360-degree annotation, and we clip the scenes into (−75.2, 75.2)m, (−75.2, 75.2)m, and (−2, 4)m for the X, Y, and Z axis ranges. We primarily focused on outdoor scene datasets whose target objects are occlusion-free in the BEV.

TABLE 1. Performance comparison on the KITTI-*val* set. The results were evaluated by the AP with 11 recall positions, and the average values of three repeated experiments were reported for each AP.

Method	Car 3D AP (IoU=0.7)			Ped 3D AP (IoU=0.5)			Cyc 3D AP (IoU=0.5)			mAP	Latency
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard		
Anchor-based Detectors											
PointPillars [15]	86.63	76.83	75.04	**55.86**	**50.18**	**46.45**	80.62	62.71	59.24	65.96	50.3ms
PointPillars + MD3D	**87.78**	**77.37**	**75.33**	53.47	44.90	42.40	**84.30**	**67.65**	**63.41**	**66.29**	**47.0ms**
SECOND [7]	88.13	78.34	77.10	55.45	51.28	47.09	80.49	65.98	61.62	67.28	21.9ms
SECOND + MD3D	**89.12**	**78.85**	**77.18**	**64.41**	**57.07**	**50.79**	**87.00**	**72.10**	**65.66**	**71.35**	**21.5ms**
PV-RCNN [12]	**89.31**	**83.04**	**78.76**	64.65	57.96	53.68	85.50	**71.61**	**68.14**	72.52	90.1ms
PV-RCNN + MD3D	89.09	81.33	78.41	**65.33**	**58.87**	**54.70**	**86.29**	71.55	67.47	**72.56**	**89.9ms**
Anchor-free Detectors											
CenterPoint [17]	85.25	77.45	76.09	56.72	52.79	49.66	82.71	67.43	62.97	67.90	28.1ms
CenterPoint + MD3D	**89.10**	**78.88**	**77.60**	**63.42**	**55.96**	**50.02**	**86.93**	**70.35**	**65.53**	**70.86**	**26.7ms**
PointRCNN [5]	**88.14**	**78.12**	**77.43**	66.79	61.90	56.29	86.33	71.83	**68.93**	72.86	135.38ms
PointRCNN + MD3D	86.75	77.17	75.65	**70.59**	**62.61**	**57.14**	**86.54**	**72.27**	68.50	**73.02**	**105.49ms**

TABLE 2. Performance comparison on the Waymo open dataset with 202 validation sequences.

Difficulty	Method	Veh AP	Veh APH	Ped AP	Ped APH	Cyc AP	Cyc APH
LEVEL_1	SECOND	67.46	66.87	57.16	47.56	55.76	54.40
	SECOND + MD3D	**69.27**	**68.51**	**63.70**	**55.80**	**67.61**	**66.22**
	Improvements	*+1.81*	*+1.64*	*+6.54*	*+8.24*	*+11.85*	*+11.82*
LEVEL_2	SECOND	59.12	58.59	49.38	41.03	53.87	52.55
	SECOND + MD3D	**60.74**	**60.05**	**54.43**	**47.58**	**65.31**	**63.97**
	Improvements	*+1.62*	*+1.46*	*+5.05*	*+6.55*	*+11.44*	*+11.42*

B. EXPERIMENT SETTINGS

We conducted the experiments with the same factors as the existing 3D object detectors, except that the detection head was replaced with MD3D. Most of the configurations are from OpenPCDet [39], one of the most commonly used codebases for 3D object detection. The detailed network structures of each detector are shown in Tables 7 and 8. The baseline detectors may differ slightly in performance owing to the gap between the settings of the original paper. As MD3D is plugged and played with the existing detectors, the only modification in the MD3D experimental setting is the detection head for one-stage detectors, and RPN head for two-stage detectors.

C. RESULTS ON KITTI DATASET

As shown in Table 1, we conducted the KITTI-*val* dataset experiment to compare the performance and speed of the baselines, three anchor-based detectors, and two anchor-free detectors. We mark '+MD3D' when the proposed method, MD3D, is applied. We calculated the average precision (AP) by creating a precision-recall curve along with changes to the confidence threshold and averaging the precision values at 11 recall points. The IoU thresholds for 3D AP were set to 0.7, 0.5, and 0.5 for Car, Pedestrian, and Cyclist, respectively, and mAP is the mean 3D AP score for all classes. Latency was measured as the total inference time, including post-processing with a batch size of 1, using Titan RTX.

MD3D improves the performance of all anchor-based detectors for most classes and difficulty levels. In the case of SECOND, a significant improvement was achieved in all settings, especially in the Pedestrian and Cyclist classes. This difference in performance gain arises from the difference in the feature dimension size. Unlike PointPillars, which use 248×216 features, SECOND uses 200×176-size features. In other words, PointPillars has a lower anchor stride than SECOND, so its anchor boxes have already been excessively assigned as a foreground for even small-size GTs of Pedestrian and Cyclist, which enables sufficient learning for both classes at the cost of latency. As a result, even if MD3D was applied, a significant performance improvement would not be achieved. However, with a larger anchor stride, SECOND assigns an average of 1.4 and 1.6 anchors to Pedestrian and Cyclist, respectively, so they are not trained sufficiently. Therefore, with MD3D learning a single GMM regardless of the size of the object, SECOND + MD3D significantly improves the performance for Pedestrian and Cyclist compared to SECOND. The performance improvement of PV-RCNN is marginal because MD3D is applied only to the RPN of the first stage. Regardless of the precision, in the RPN, the recall value increases with the increase in number of proposal boxes. However, as can be seen in Fig. 7, MD3D is more effective at removing false positives than the existing head; therefore, the performance improvement of an RPN head is insufficient.

The MD3D also shows superior performance and latency compared to anchor-free detection heads. Compared to CenterPoint [17], which uses a heatmap-based detection head and regresses the bounding box with center coordinates, MD3D significantly improves the performance for all classes and reduces latency. This performance improvement is inherently attributed to MD3D's effective learning for small classes,

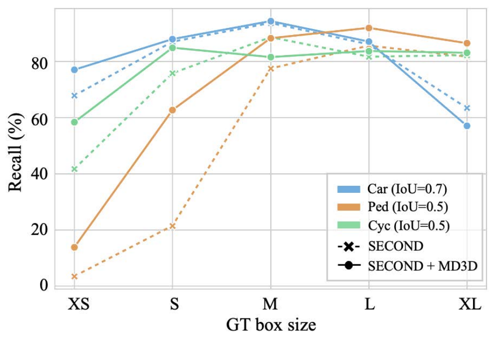


FIGURE 4. Recalls on the KITTI-*val* set. MD3D offers a clear advantage in predicting small objects, where only a limited number of anchor boxes are assigned to the existing detectors.

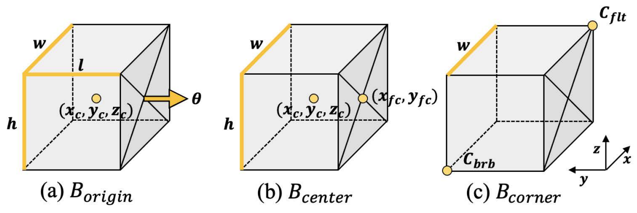


FIGURE 5. Illustration of various box encoding methods. The B_{origin} consists of the center coordinate (x_c, y_c, z_c), the dimension (l, w, h), and the yaw angle θ. The B_{center} consists of the center coordinate (x_c, y_c, z_c), the front-center coordinate (x_{fc}, y_{fc}), and the dimension (w, h). We use B_{corner} consisting of the front-left-top corner $C_{flt} = (x_{flt}, y_{flt}, z_{flt})$, the back-right-bottom corner $C_{brb} = (x_{brb}, y_{brb}, z_{brb})$, and width w.

such as Pedestrian and Cyclist, in addition to the change of box encoding scheme from center to corner. The Gaussian radius of the heatmap depends on the size of the GT box; therefore, small objects are not sufficiently trained as large ones. For another anchor-free detector, PointRCNN [5], which is a two-stage detector that utilizes point features, we replace only the regression branch with MD3D while leaving the classification branch that performs foreground segmentation in the RPN as it is. With this modification, the mAP is increased slightly, and the latency was significantly reduced. This shows that our MDN-based corner regression offers an advantage over PointRCNN's bin-based residual regression. The latency of our method is significantly reduced because unnecessary boxes are removed using ϕ before the NMS. For both the training and inference phases, we keep the top 512 proposals for refinement of the stage-2 sub-network.

D. RESULTS ON WAYMO OPEN DATASET

For the Waymo dataset, we report the AP and the average precision weighted by heading (APH) of SECOND [7] with and without the MD3D head, respectively. The AP was calculated by averaging the precision values at 11 recall points identically to that of the KITTI dataset. APH is calculated similarly to AP but uses precision values weighting each true positive by heading accuracy. We evaluated the models into two difficulty levels: LEVEL 1 includes GT boxes with at least five inside points, and LEVEL 2 includes GTs with at least one inside point. As shown in Table 2, the proposed MD3D improved the baseline at all levels and all classes. Note that the MD3D leads to a significant gain in Pedestrian and Cyclist classes, whose GTs are relatively small. This verifies the advantages of the MD3D predicting bounding boxes in a probabilistic and anchor-free manner.

E. ANALYZING RECALL BY OBJECT SIZE

As shown in Fig. 1, anchor-based detectors have an insufficient number of anchors assigned to the foreground for extremely small objects. We measured recall by object size to verify that the proposed method, not in the use of anchors, could improve this inherent limitation. We used SECOND as the base model, and considered bounding boxes before NMS to focus on the regression results. As shown in Fig. 4, there is an improvement for XS-sized GT boxes in the Car class, where the lack of assigned anchors has caused harm to the performance of the existing detector. The improvement is insignificant for other sized Car boxes because the base model has already assigned an excessive number of anchors to the foreground. In addition, in Pedestrian and Cyclist classes, the recall of GT with a size close to the average is already high enough for the base model, so the increase is small; however, for extremely small size cases, the increase is noticeably significant. Therefore, the proposed MD3D can delicately detect small objects compared to anchor-based detection heads.

F. ABLATION STUDY

1) BOX ENCODING METHODS

As shown in Fig. 5, we conducted an ablation study of the box encoding method on the KITTI validation dataset with models applying MD3D to PointPillars. To better demonstrate the performance of each box encoding method regarding the headings of predicted boxes, we used the average orientation similarity (AOS) metric, along with 3D AP, which assesses cosine similarities between the angles of the estimated and GT heading orientations. The results are presented in Table 3. First, using the GT box in its original form, $B_{origin} = \{x_c, y_c, z_c, l, w, h, \theta\}$ results in a very low AOS AP and thus a low 3D AP. This is because of discontinuous θ, which is the same box when the yaw angle θ is 0 and 2π, but the loss is calculated differently. Therefore, we experimented with B_{sincos}, which changes θ to a continuous value, $(sin(\theta), cos(\theta)) \in \mathbb{R}^2$, to avoid ambiguity. Both AOS and 3D AP have some increases. Still, because $sin(\theta)$ and $cos(\theta)$ are mutually dependent and periodic, B_{sincos} is not entirely appropriate for our GMM modeling, leaving room for improvement. Therefore we devised a novel method of finding the front-center coordinate value of the box, as shown in (b), to predict the box without θ. This $B_{center} = \{x_c, y_c, z_c, x_{fc}, y_{fc}, w, h\}$ has a notable improvement over the previous two approaches. In addition, another variant method, B_{corner}, predicting two corners with w, yields the highest performance. This is because it is easy to localize the corner of an object owing to the characteristics of the point clouds obtained by LiDAR.

TABLE 3. Comparison of box encoding methods.

Box encoding	AOS AP_{Hard}		3D AP_{Hard}	
	Ped	Cyc	Ped	Cyc
B_{origin}	31.89	64.29	32.70	59.91
B_{sincos}	31.69	67.03	36.90	61.93
B_{center}	**39.51**	68.02	41.88	62.61
B_{corner}	39.31	**71.13**	**42.40**	**63.41**

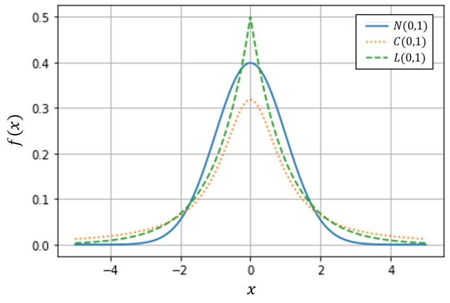


FIGURE 6. Pdfs of the Gaussian, Cauchy, and laplace distributions.

TABLE 4. Comparison of probability distributions.

PDF form	KITTI-*val* 3D mAP	
	PointPillars+MD3D	SECOND+MD3D
Laplace	63.98	70.47
Cauchy	64.91	70.75
Gaussian	**66.29**	**71.35**

2) PROBABILITY DISTRIBUTIONS

In MD3D, it is essential to choose a proper probability density function that fits the data characteristics of the input point clouds and the output 3D bounding boxes, because it substantially impacts the model's performance. We consider the Laplace, Cauchy, and Gaussian distributions, which are symmetric and have the same number of parameters. We applied them to PointPillars and SECOND baselines and compared them to the KITTI-*val* dataset. As shown in Fig. 6, the shapes of the three distributions differ in peak height and tail length. The point clouds have sparse representations, implying that the area occupied by actual points is very small compared with the space of the entire area when voxelized. This makes our MD3D have a considerable number of mixture components because MD3D requires output for the entire feature space. Therefore, the Gaussian distribution with the shortest tail is the most suitable for MD3D because it can effectively suppress the probability of boxes with high uncertainty from unnecessary empty spaces. Table 4 also shows that the Gaussian distribution was the most effective for both models.

3) THE NUMBER OF MIXTURE COMPONENTS

Table 1 shows that MD3D has a significant performance improvement for detectors with smaller input feature dimension and therefore fewer anchors. To verify the effect of the number of anchors and the number of mixture components K, we set SECOND as the base model and compared the performance by adjusting the horizontal and vertical dimensions of the input feature by 1/4, 1/2, and 2 times, respectively. As shown in Table 5, anchors are placed separately in two directions, 0 and 90 degrees for each class. Thereby, anchor-based detectors output anchor boxes six times more than K, even for features of the same size. The anchor-based detector with more anchors achieves higher performance in obtaining better foreground GT assignments, whereas MD3D with an unnecessarily large K tends to learn poorly and achieve lower performance. However, comparing them in a small feature dimension of 50×44, SECOND achieves a very low mAP (50.10) despite predicting six times more boxes than MD3D (64.86). SECOND needs to predict $200 \times 176{\times}6$ boxes, 96 times more boxes than MD3D, to achieve similar performance (67.28). Therefore, the performance of MD3D is maximized when used in compact detectors.

TABLE 5. Effect of the number of mixture components.

Method	# anchors	KITTI-*val* 3D AP_{mean}			mAP
	K	Car	Ped	Cyc	
SECOND	50×44×6	66.12	29.91	54.27	50.10
	100×88×6	78.78	33.37	62.51	58.22
	200×176×6	**81.19**	51.27	69.36	67.28
	400×352×6	80.28	**63.58**	**72.98**	**72.28**
SECOND +MD3D	50×44	72.94	49.85	71.78	64.86(*+14.76*)
	100×88	81.10	55.05	74.45	70.20(*+11.98*)
	200×176	**81.72**	57.42	**74.92**	**71.35**(*+ 4.08*)
	400×352	81.07	**57.46**	74.46	71.00(*- 1.28*)

TABLE 6. Effect of constraint on the covariance matrix. Assuming independence between elements achieves relatively efficient and effective results.

Covariance matrix	KITTI-*val* 3D AP_{mean} Car	Ped	Cyc	mAP	MD3D params
Full	75.06	**53.12**	68.35	65.51	15,018
Diagonal	**80.16**	46.92	**71.79**	**66.29**	**6,930**

4) COVARIANCE MATRIX

We modeled the point cloud-based 3D object detection with the multivariate GMM using only the diagonal elements of a covariance matrix rather than a full matrix. Training with a full covariance matrix means that a detector learns the correlation of the elements of $B_{corner} \in \mathbb{R}^7$, whereas the diagonal matrix does not. Table 6 presents the results of applying these two methods to PointPillars. In the case of Pedestrian, whose intraclass correlation is high because the objects share similar shapes, the full covariance model achieves higher performance. Except for Pedestrian, the models using only the diagonal matrix outperformed those using the entire matrix. Therefore, we decided to use only the diagonal elements of the covariance matrix because it achieves a slightly better mAP and uses half the number of parameters.

G. DISCUSSION

MD3D showed superior performance and latency regardless of the backbone network types and the use of anchors

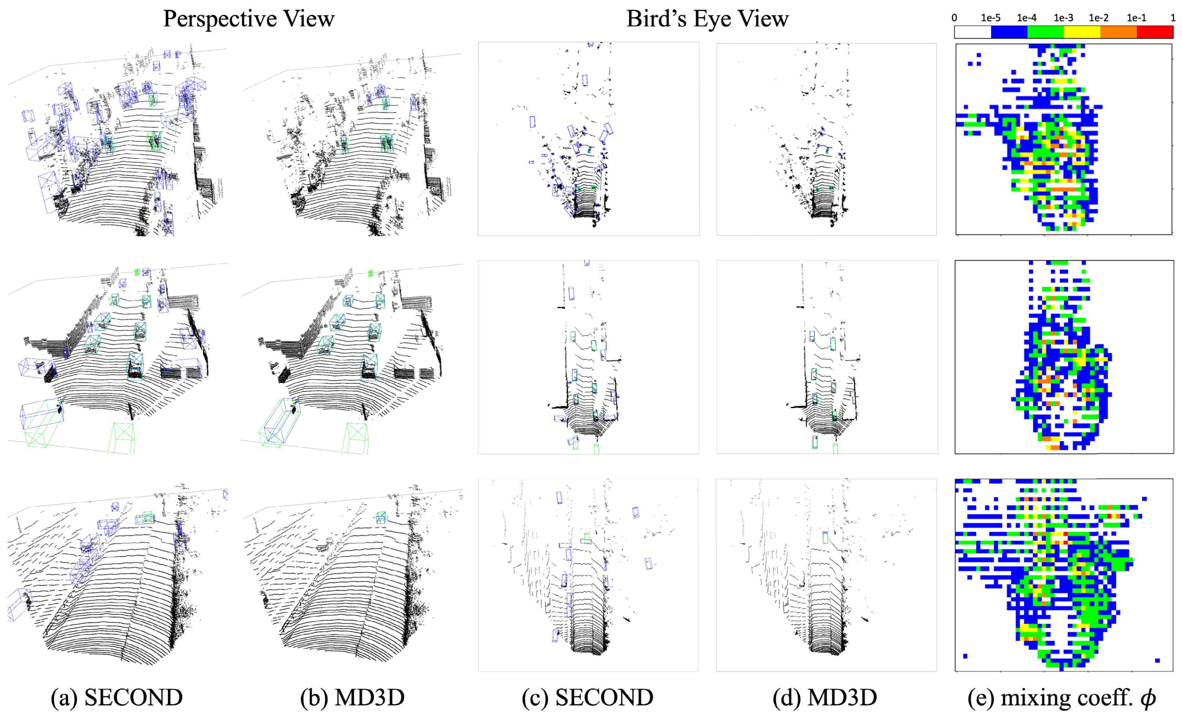
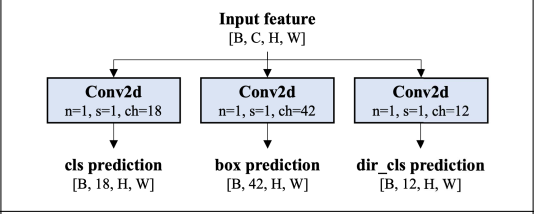
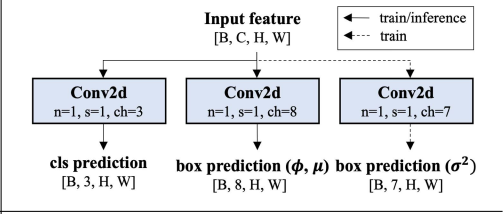
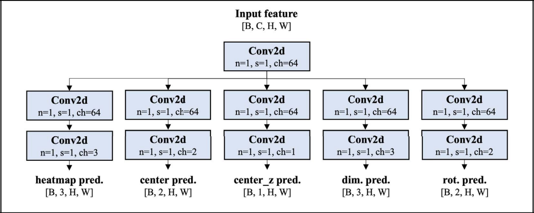
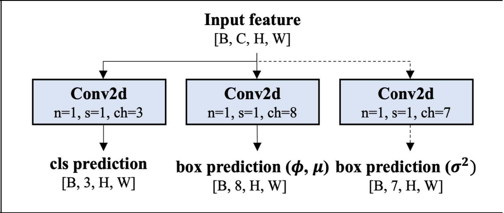
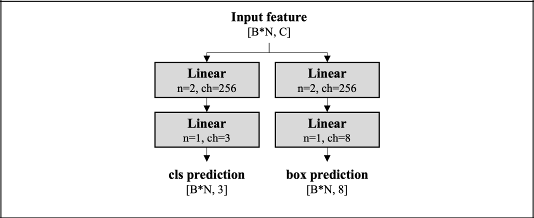
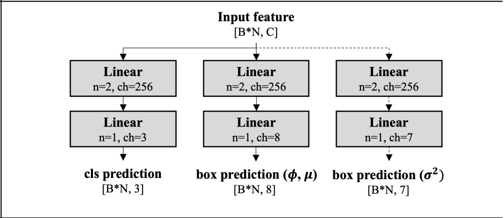


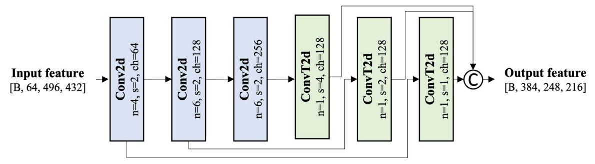
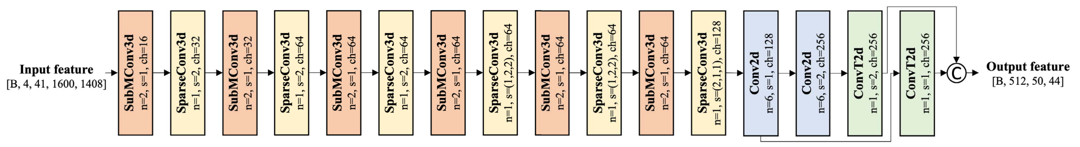
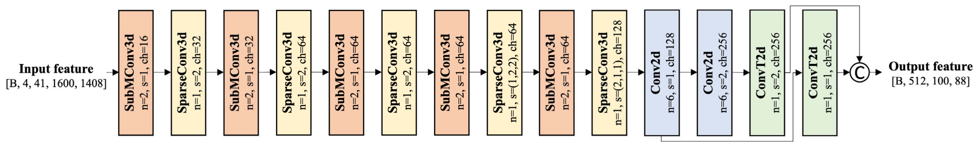
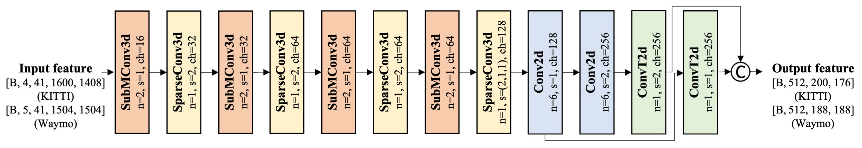
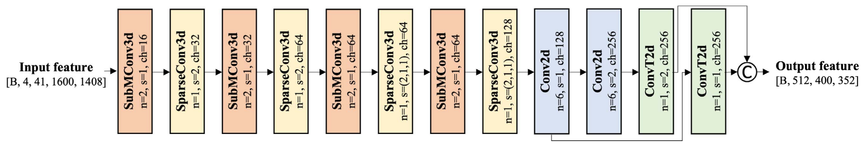
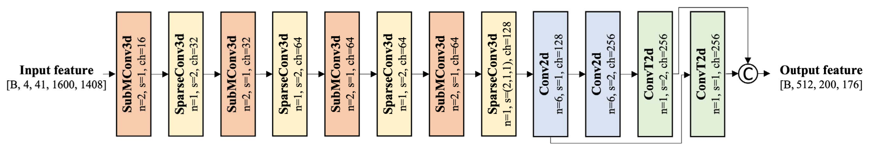
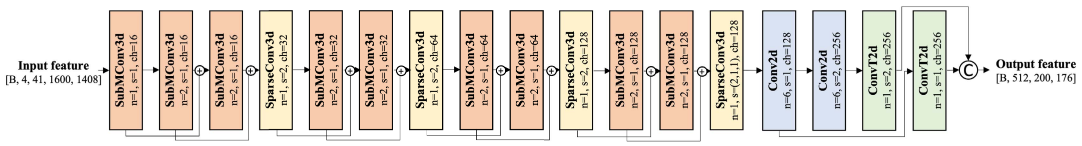
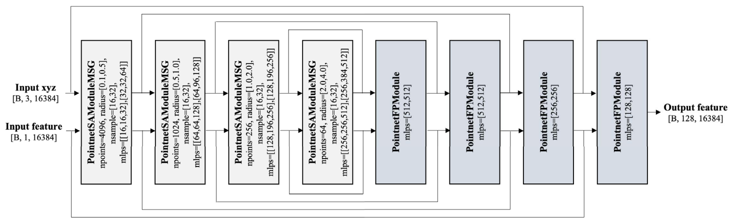
FIGURE 7. Qualitative results for the KITTI-*val* set. Green boxes indicate GT boxes, and blue boxes indicate predicted boxes with the confidence of over 0.1. MD3D improves the existing detector regarding precision and recall, whereas the mixing coefficient ϕ efficiently filters out unlikely boxes. We filtered out the predictions with $\frac{\phi}{max(\phi)} < 0.001$ before the NMS. We used SECOND with a 50×44 feature as the base model for this comparison.

TABLE 7. Detection heads. The structures and parameters of the original heads are the default settings of OpenPCDet [39]. The structures and parameters of the MD3D BEV-heads (PointPillars, SECOND, PV-RCNN, and CenterPoint) follow the simplest original BEV head and the MD3D point-head (PointRCNN) follows the original PointRCNN head. σ^2 (dotted line) is not used in the inference phase.

Method	Detection head	
	Original	MD3D
PointPillars SECOND PV-RCNN	 Input feature [B, C, H, W] Conv2d n=1, s=1, ch=18 Conv2d n=1, s=1, ch=42 Conv2d n=1, s=1, ch=12 cls prediction [B, 18, H, W] box prediction [B, 42, H, W] dir_cls prediction [B, 12, H, W]	 Input feature [B, C, H, W] train/inference train Conv2d n=1, s=1, ch=3 Conv2d n=1, s=1, ch=8 Conv2d n=1, s=1, ch=7 cls prediction [B, 3, H, W] box prediction (ϕ, μ) [B, 8, H, W] box prediction (σ^2) [B, 7, H, W]
CenterPoint	 Input feature [B, C, H, W] Conv2d n=1, s=1, ch=64 Conv2d n=1, s=1, ch=64 Conv2d n=1, s=1, ch=64 Conv2d n=1, s=1, ch=64 Conv2d n=1, s=1, ch=64 Conv2d n=1, s=1, ch=64 Conv2d n=1, s=1, ch=3 Conv2d n=1, s=1, ch=2 Conv2d n=1, s=1, ch=1 Conv2d n=1, s=1, ch=3 Conv2d n=1, s=1, ch=2 heatmap pred. [B, 3, H, W] center pred. [B, 2, H, W] center_z pred. [B, 1, H, W] dim. pred. [B, 3, H, W] rot. pred. [B, 2, H, W]	 Input feature [B, C, H, W] Conv2d n=1, s=1, ch=3 Conv2d n=1, s=1, ch=8 Conv2d n=1, s=1, ch=7 cls prediction [B, 3, H, W] box prediction (ϕ, μ) [B, 8, H, W] box prediction (σ^2) [B, 7, H, W]
PointRCNN	 Input feature [B*N, C] Linear n=2, ch=256 Linear n=2, ch=256 Linear n=1, ch=3 Linear n=1, ch=8 cls prediction [B*N, 3] box prediction [B*N, 8]	 Input feature [B*N, C] Linear n=2, ch=256 Linear n=2, ch=256 Linear n=2, ch=256 Linear n=1, ch=3 Linear n=1, ch=8 Linear n=1, ch=7 cls prediction [B*N, 3] box prediction (ϕ, μ) [B*N, 8] box prediction (σ^2) [B*N, 7]

(Tables 1 and 2). It is especially effective for one-stage detectors, such as SECOND and CenterPoint, which output relatively small feature maps (Table 5). The reason for their dramatic increase in performance is the higher recall than that of existing heads for small objects, such as Pedestrian and Cyclist (Fig. 4). However, MD3D has an advan-

TABLE 8. Backbone network architectures. The backbone structures of SECOND(50 × 44, 100 × 88, and 400 × 352) were designed by adding some sparse convolution blocks or changing the stride parameters of the default structure of SECOND(200 × 176). Other models used the default settings of OpenPCDet [39].

Method	Backbone network
PointPillars	
SECOND (50×44)	
SECOND (100×88)	
SECOND (200×176)	
SECOND (400×352)	
PV-RCNN	
CenterPoint	
PointRCNN	

Input feature [B, 64, 496, 432]
Conv2d n=4, s=2, ch=64
Conv2d n=6, s=2, ch=128
Conv2d n=6, s=2, ch=256
ConvT2d n=1, s=4, ch=128
ConvT2d n=1, s=2, ch=128
ConvT2d n=1, s=1, ch=128
C
Output feature [B, 384, 248, 216]

Input feature [B, 4, 41, 1600, 1408]
SubMConv3d n=2, s=1, ch=16
SparseConv3d n=1, s=2, ch=32
SubMConv3d n=2, s=1, ch=32
SparseConv3d n=1, s=2, ch=64
SubMConv3d n=2, s=1, ch=64
SparseConv3d n=1, s=2, ch=64
SubMConv3d n=2, s=1, ch=64
SparseConv3d n=1, s=(1,2,2), ch=64
SubMConv3d n=2, s=1, ch=64
SparseConv3d n=1, s=(1,2,2), ch=64
SubMConv3d n=2, s=1, ch=64
SparseConv3d n=1, s=(2,1,1), ch=128
Conv2d n=6, s=1, ch=128
Conv2d n=6, s=2, ch=256
ConvT2d n=1, s=2, ch=256
ConvT2d n=1, s=1, ch=256
C
Output feature [B, 512, 50, 44]

Input feature [B, 4, 41, 1600, 1408]
SubMConv3d n=2, s=1, ch=16
SparseConv3d n=1, s=2, ch=32
SubMConv3d n=2, s=1, ch=32
SparseConv3d n=1, s=2, ch=64
SubMConv3d n=2, s=1, ch=64
SparseConv3d n=1, s=2, ch=64
SubMConv3d n=2, s=1, ch=64
SparseConv3d n=1, s=(1,2,2), ch=64
SubMConv3d n=2, s=1, ch=64
SparseConv3d n=1, s=(2,1,1), ch=128
Conv2d n=6, s=1, ch=128
Conv2d n=6, s=2, ch=256
ConvT2d n=1, s=2, ch=256
ConvT2d n=1, s=1, ch=256
C
Output feature [B, 512, 100, 88]

Input feature [B, 4, 41, 1600, 1408] (KITTI) [B, 5, 41, 1504, 1504] (Waymo)
SubMConv3d n=2, s=1, ch=16
SparseConv3d n=1, s=2, ch=32
SubMConv3d n=2, s=1, ch=32
SparseConv3d n=1, s=2, ch=64
SubMConv3d n=2, s=1, ch=64
SparseConv3d n=1, s=2, ch=64
SubMConv3d n=2, s=1, ch=64
SparseConv3d n=1, s=(2,1,1), ch=128
Conv2d n=6, s=1, ch=128
Conv2d n=6, s=2, ch=256
ConvT2d n=1, s=2, ch=256
ConvT2d n=1, s=1, ch=256
C
Output feature [B, 512, 200, 176] (KITTI) [B, 512, 188, 188] (Waymo)

Input feature [B, 4, 41, 1600, 1408]
SubMConv3d n=2, s=1, ch=16
SparseConv3d n=1, s=2, ch=32
SubMConv3d n=2, s=1, ch=32
SparseConv3d n=1, s=2, ch=64
SubMConv3d n=2, s=1, ch=64
SparseConv3d n=1, s=(2,1,1), ch=64
SubMConv3d n=2, s=1, ch=64
SparseConv3d n=1, s=(2,1,1), ch=128
Conv2d n=6, s=1, ch=128
Conv2d n=6, s=2, ch=256
ConvT2d n=1, s=2, ch=256
ConvT2d n=1, s=1, ch=256
C
Output feature [B, 512, 400, 352]

Input feature [B, 4, 41, 1600, 1408]
SubMConv3d n=2, s=1, ch=16
SparseConv3d n=1, s=2, ch=32
SubMConv3d n=2, s=1, ch=32
SparseConv3d n=1, s=2, ch=64
SubMConv3d n=2, s=1, ch=64
SparseConv3d n=1, s=2, ch=64
SubMConv3d n=2, s=1, ch=64
SparseConv3d n=1, s=(2,1,1), ch=128
Conv2d n=6, s=1, ch=128
Conv2d n=6, s=2, ch=256
ConvT2d n=1, s=2, ch=256
ConvT2d n=1, s=1, ch=256
C
Output feature [B, 512, 200, 176]

Input feature [B, 4, 41, 1600, 1408]
SubMConv3d n=1, s=1, ch=16
SubMConv3d n=2, s=1, ch=16
SubMConv3d n=2, s=1, ch=16
SparseConv3d n=1, s=2, ch=32
SubMConv3d n=2, s=1, ch=32
SubMConv3d n=2, s=1, ch=32
SparseConv3d n=1, s=2, ch=64
SubMConv3d n=2, s=1, ch=64
SubMConv3d n=2, s=1, ch=64
SparseConv3d n=1, s=2, ch=128
SubMConv3d n=2, s=1, ch=128
SubMConv3d n=2, s=1, ch=128
SparseConv3d n=1, s=(2,1,1), ch=128
Conv2d n=6, s=1, ch=128
Conv2d n=6, s=2, ch=256
ConvT2d n=1, s=2, ch=256
ConvT2d n=1, s=1, ch=256
C
Output feature [B, 512, 200, 176]

Input xyz [B, 3, 16384]
Input feature [B, 1, 16384]
PointnetSAModuleMSG npoints=4096, radius=[0.1,0.5], nsample=[16,32], mlps=[[16,16,32],[32,32,64]]
PointnetSAModuleMSG npoints=1024, radius=[0.5,1.0], nsample=[16,32], mlps=[[64,64,128],[64,96,128]]
PointnetSAModuleMSG npoints=256, radius=[1.0,2.0], nsample=[16,32], mlps=[[128,196,256],[128,196,256]]
PointnetSAModuleMSG npoints=64, radius=[2.0,4.0], nsample=[16,32], mlps=[[256,256,512],[256,384,512]]
PointnetFPModule mlps=[512,512]
PointnetFPModule mlps=[512,512]
PointnetFPModule mlps=[256,256]
PointnetFPModule mlps=[128,128]
Output feature [B, 128, 16384]

tage in terms of recall rather than precision (Fig. 7), the performance improvement for two-stage detectors, such as PV-RCNN and PointRCNN is marginal. In addition, MD3D has advantages regarding the number of parameters and latency because the box prediction channels are not separated by class. However, because of the unified channel across classes, the performance on datasets with many classes may be limited, which we will attempt to overcome in the future work.

V. CONCLUSION

Most of the existing point cloud-based 3D object detectors apply a specific target assignment policy to the GT boxes to regress 3D bounding boxes. Because this training method needs to optimize many hand-crafted design factors, it takes significant amount of effort to utilize and places many restrictions on the network structure. In this paper, we proposed MD3D, which reformulates the regression of 3D bounding boxes in point clouds as a density estimation problem. The MD3D is easy to use and can be applied to various types of feature encoding methods without considering the target assignment policy and network structure. Experiments on the KITTI and Waymo datasets show that the proposed method outperforms conventional methods in terms of performance, speed, ease of use, and flexibility. Although we only considered point clouds as inputs, the MD3D can be easily applied to other various types of inputs. Furthermore, we expect MD3D is utilized for multi-modal inputs by fusing the mixture density outputs.

REFERENCES

[1] J. Choi, Y. Song, and N. Kwak, "Part-aware data augmentation for 3D object detection in point cloud," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2021, pp. 3391–3397.

[2] Q. Xu, Y. Zhong, and U. Neumann, "Behind the curtain: Learning occluded shapes for 3D object detection," 2021, *arXiv:2112.02205*.

[3] 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.*, Jul. 2017, pp. 652–660.

[4] 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.*, 2017, pp. 5099–5108.

[5] 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.

[6] I. Misra, R. Girdhar, and A. Joulin, "An end-to-end transformer model for 3D object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Oct. 2021, pp. 2906–2917.

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

[8] W. Zheng, W. Tang, S. Chen, L. Jiang, and C.-W. Fu, "CIA-SSD: Confident IoU-aware single-stage object detector from point cloud," 2020, *arXiv:2012.03015*.

[9] H. Kuang, B. Wang, J. An, M. Zhang, and Z. Zhang, "Voxel-FPN: Multi-scale voxel feature aggregation for 3D object detection from LIDAR point clouds," *Sensors*, vol. 20, no. 3, p. 704, 2020.

[10] 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.*, 2021, vol. 35, no. 2, pp. 1201–1209.

[11] W. Shi and R. Rajkumar, "Point-GNN: Graph neural network for 3D object detection in a point cloud," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2020, pp. 1711–1719.

[12] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li, "PV-RCNN: Point-voxel feature set abstraction for 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2020, pp. 10529–10538.

[13] C. He, H. Zeng, J. Huang, X.-S. Hua, and L. Zhang, "Structure aware single-stage 3D object detection from point cloud," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2020, pp. 11873–11882.

[14] 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.*, Oct. 2021, pp. 2723–2732.

[15] 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 Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 12697–12705.

[16] 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)*, Nov. 2019, pp. 1951–1960.

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

[18] R. Ge, Z. Ding, Y. Hu, Y. Wang, S. Chen, L. Huang, and Y. Li, "AFDet: Anchor free one stage 3D object detection," 2020, *arXiv:2006.12671*.

[19] Y. Hu, Z. Ding, R. Ge, W. Shao, L. Huang, K. Li, and Q. Liu, "AFDetV2: Rethinking the necessity of the second stage for object detection from point clouds," 2021, *arXiv:2112.09205*.

[20] Q. Chen, L. Sun, Z. Wang, K. Jia, and A. Yuille, "Object as hotspots: An anchor-free 3D object detection approach via firing of hotspots," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 68–84.

[21] J. Li, H. Dai, L. Shao, and Y. Ding, "Anchor-free 3D single stage detector with mask-guided attention for point cloud," in *Proc. 29th ACM Int. Conf. Multimedia*, 2021, pp. 553–562.

[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] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 2980–2988.

[24] C. R. Qi, O. Litany, K. He, and L. J. Guibas, "Deep Hough voting for 3D object detection in point clouds," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Oct. 2019, pp. 9277–9286.

[25] F. Su, H. Zhu, T. Chen, L. Li, F. Yang, H. Peng, L. Tang, X. Zuo, Y. Liang, and S. Ying, "An anchor-based graph method for detecting and classifying indoor objects from cluttered 3D point clouds," *ISPRS J. Photogramm. Remote Sens.*, vol. 172, pp. 114–131, Feb. 2021.

[26] Y. Chen, Y. Li, X. Zhang, J. Sun, and J. Jia, "Focal sparse convolutional networks for 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2022, pp. 5428–5437.

[27] 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.

[28] L. Fan, Z. Pang, T. Zhang, Y.-X. Wang, H. Zhao, F. Wang, N. Wang, and Z. Zhang, "Embracing single stride 3D object detector with sparse transformer," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2022, pp. 8458–8468.

[29] H. Law and J. Deng, "CornerNet: Detecting objects as paired keypoints," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 734–750.

[30] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," 2019, *arXiv:1904.07850*.

[31] Z. Tian, C. Shen, H. Chen, and T. He, "FCOS: Fully convolutional one-stage object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Oct. 2019, pp. 9627–9636.

[32] C. M. Bishop, "Mixture density networks," Aston Univ., Birmingham, U.K., Tech. Rep. NCRG/94/004, 1994, p. 26.

[33] Y. He and J. Wang, "Deep mixture density network for probabilistic object detection," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 10550–10555.

[34] D. Feng, L. Rosenbaum, and K. Dietmayer, "Towards safe autonomous driving: Capture uncertainty in the deep neural network for lidar 3D vehicle detection," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 3266–3273.

[35] A. Varamesh and T. Tuytelaars, "Mixture dense regression for object detection and human pose estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 13086–13095.

[36] J. Choi, I. Elezi, H.-J. Lee, C. Farabet, and J. M. Alvarez, "Active learning for deep object detection via probabilistic modeling," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 10264–10273.

[37] J. Yoo, H. Lee, I. Chung, G. Seo, and N. Kwak, "Training multi-object detector by estimating bounding box distribution for input image," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Oct. 2021, pp. 3437–3446.

[38] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, and V. Vasudevan, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 2446–2454.

[39] Organization Development Team. (2020). *OpenPCDet: An Open-Source Toolbox for 3D Object Detection From Point Clouds*. [Online]. Available: https://github.com/open-mmlab/OpenPCDet

JAESEOK CHOI was born in Daegu, South Korea, in 1994. He received the B.S. degree in computer engineering from Sungkyunkwan University, Seoul, South Korea, in 2017. He is currently pursuing the Ph.D. degree with the Department of Intelligence and information, Seoul National University, Seoul. His research interest includes 2D/3D object detection for autonomous vehicles.

YEJI SONG received the B.S. degree in statistics and psychology (double major) from Seoul National University, Seoul, South Korea, in 2020, where she is currently pursuing the Ph.D. degree with the Department of Intelligence and Information. Her research interests include 3D scene rendering and 3D object detection.

YERIM KIM received the B.S. degree in statistics and economics (double major) from Korea University, Seoul, South Korea, in 2021. She is currently pursuing the master's degree with the Department of Intelligence and Information, Seoul National University, Seoul. Her research interests include computer vision and explainable AI.

JAEYOUNG YOO received the B.S. degree in computer science from Soongsil University, Seoul, South Korea, in 2015, and the Ph.D. degree in intelligence systems from Seoul National University, Seoul, in 2021. Currently, he is an AI Researcher with NAVER WEBTOON, Seongnam, South Korea. His research interests include multi-object detection, low-level vision, and generative model.

NOJUN KWAK (Senior Member, IEEE) was born in Seoul, South Korea, in 1974. He received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer science from Seoul National University, Seoul, in 1997, 1999, and 2003, respectively. From 2003 to 2006, he was at Samsung Electronics, Seoul. In 2006, he joined Seoul National University as a BK21 Assistant Professor. From 2007 to 2013, he was a Faculty Member at the Department of Electrical and Computer Engineering, Ajou University, Suwon, South Korea. Since 2013, he has been with the Graduate School of Convergence Science and Technology, Seoul National University, where he is currently a Professor. His current research interests include feature learning by deep neural networks and their applications in various areas of pattern recognition, computer vision, and image processing.