2022 Waymo 3D Camera-Only Detection Challenge Report - CMKD

Yu Hong [1] Hang Dai[2] Xu Liu[3] Yong Ding[1] Andy Wang[3] Hongyang Zhang[3] Guangzhi Cao[3]

[1]Zhejiang University
[2]Mohamed bin Zayed University of Artificial Intelligence
[3]Pegasus Tech

yuhong_1999@zju.edu.cn hang.dai@mbzuai.ac.ae dingy@vlsi.zju.edu.cn

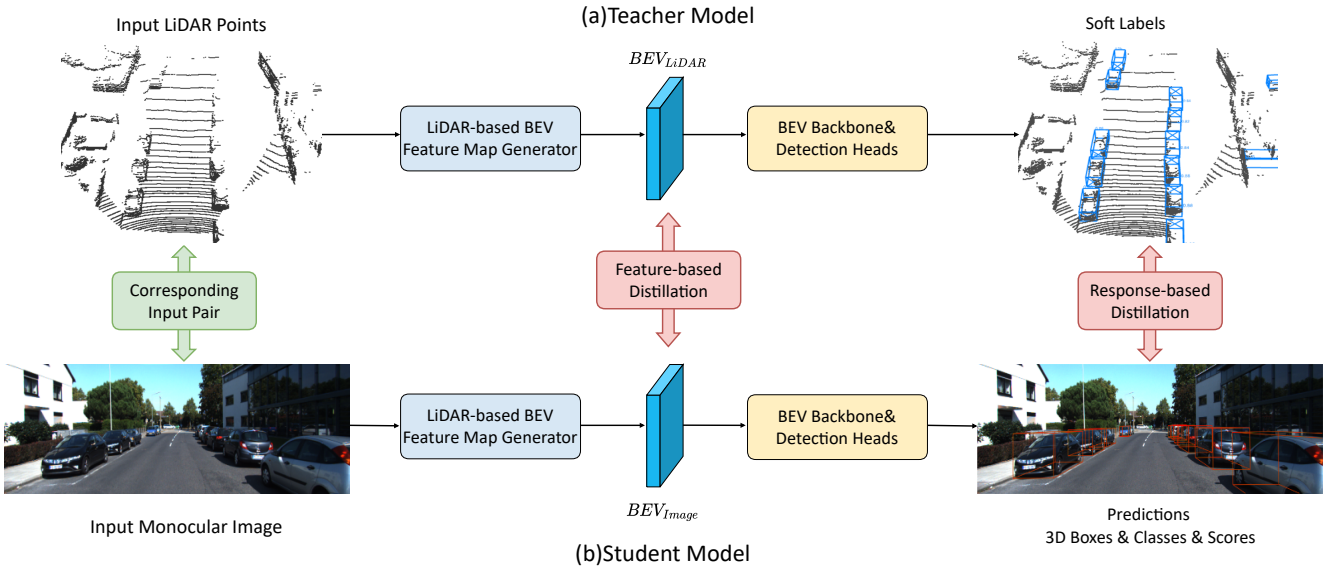

Figure 1. Overview of our cross-modality knowledge distillation framework for monocular 3D detection: (a) a pre-trained LiDAR-based 3D detector as the teacher model; (b) an image-based monocular 3D detector as the student model. The pre-trained teacher model provides distillation guidance to train the student model which is used for inference alone.

Abstract

*This is the report for the 3^{rd} ranking method **CMKD** in the 2022 Waymo 3D Camera-only Detection Challenge. In this work we propose a simple framework, namely **C**ross-**M**odality **K**nowledge **D**istillation Network, where we take advantage of LiDAR data for useful knowledge transferring to boost the performance of the Bird's-Eye-View(BEV) feature map based monocular 3D detector. Specifically, **using only 20% of the total training samples, no previous frames, no data augmentation and no training&testing tricks, CMKD ranks among the top submissions on the leaderboard**. And we believe the performance could be much further boosted with more engineering and fine-tuning. Additionally, CMKD can be also be considered as a **plug-and-play component** for a BEV-based monocular 3D detector to further boost the performance without bringing any extra cost in the inference stage.*

1. Framework Overview

Fig. 1 illustrates an overview of our cross-modality knowledge distillation (CMKD) framework for monocular 3D object detection. The key is to extract the same type of **feature** and **response** representations from point clouds and images, and then perform feature-based and response-based **knowledge distillation** between the two modalities. Our framework includes a pre-trained LiDAR-based 3D detector that provides distillation guidance used only in the training stage as the teacher model, an image-based monocular 3D detector as the student model, and the cross-modality knowledge distillation on both features and responses.

Training. In the training stage, we take the monocular images and the corresponding LiDAR point clouds as the input pair. Both the LiDAR-based teacher model and the image-base student model are composed of a Bird's-Eye-View(BEV) feature map generator to generate the BEV

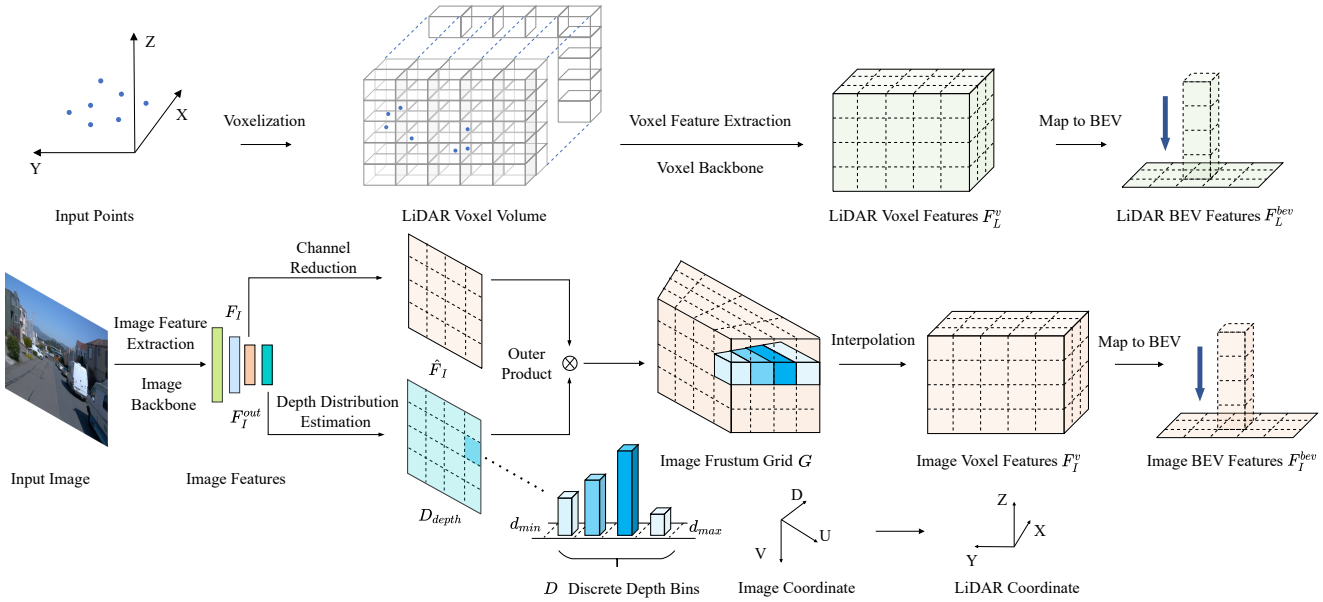

Figure 2. BEV feature map generation. The top row is the LiDAR-based branch, and the bottom row is the image-based branch.

feature map in correspondence to each modality input, and aligned BEV backbone&detection heads after the BEV feature map to perform 3D detection.

Inference. In the inference stage, we use the student model alone to perform 3D object detection with monocular images as input only.

2. BEV Feature Map Generation

2.1. LiDAR-based BEV Feature Map Generation

Voxelization. We use the LiDAR-based detector CenterPoint[1][2] [11] to generate the LiDAR BEV feature map. Given the point clouds of range (L, W, H), where L, W and H are the length, width and height of the 3D space corresponding to the X, Y and Z axis respectively, we first filter out the points that can be projected onto the image plane, $i.e.$, Field of View (FoV) points. Then, we subdivide the 3D space into equal 3D voxels of size (dx, dy, dz) to obtain the original voxel volume of size $(L/dx, W/dy, H/dz)$. Taking the Front Camera for example, the range for the point clouds in the 3D space is set to $[0, 73.6]\times[-36.8, 36.8]\times[-2.0, 4.0]$ meters. The voxel size is set to $(0.05, 0.05, 0.1)$ meters, and the size of the original voxel volume is $(1472, 1472, 60)$.

Voxel Feature Extraction. The features of the original voxels are the mean values of the point features in each voxel denoted as $\tilde{F}_L^v \in \mathbb{R}^{1472\times1472\times60\times5}$. After voxelization, $\tilde{F}_L^v$ is fed to a sparse 3D convolution backbone, gradually converting the original voxel features into higher dimensional space with 1×, 2×, 4×, 8× down-sample rates using a series of sparse 3D convolution blocks. The output features of the voxel backbone are denoted as $F_L^v \in \mathbb{R}^{184\times184\times2\times64}$.

Map to BEV. The voxel features $F_L^v \in \mathbb{R}^{184\times184\times2\times64}$ are collapsed to a 2D BEV feature map by stacking the features in height dimension to obtain the LiDAR BEV feature map $F_L^{bev} \in \mathbb{R}^{184\times184\times2*64}$.

We illustrate the BEV feature map generating process of the LiDAR-based method in the top row of Fig. 2. More details can be found in CenterPoint [11].

2.2. Image-based BEV Feature Map Generation

Image Feature Extraction. We follow CaDDN[3][4] [10] to generate the image BEV feature map. Given the monocular image $I \in \mathbb{R}^{W\times H\times3}$, where W and H are 1920 and 1280 in Waymo [4](images from side cameras are padded with zeros), we first resize the images to 1920×600, and then use a ResNet-50 [5] backbone to extract the image features. In our case, we use the intermediate image features from $layer3$ [5] with down-sample rate 8 as the image features $F_I \in \mathbb{R}^{W_I\times H_I\times C}$ where $W_I = 240$, $H_I = 75$ and $C = 1024$. We use the output image features $F_I^{out} \in \mathbb{R}^{W_I\times H_I\times C_{out}}$ where $W_I = 240$, $H_I = 75$ and $C_{out} = 2048$ as the output feature of the image backbone, which are then fed to the depth distribution estimation head described below.

Voxelization via Depth Distribution Estimation. The image features $F_I \in \mathbb{R}^{W_I\times H_I\times C}$ go through a channel re-

[1] https://github.com/tianweiy/CenterPoint
[2] https://github.com/open-mmlab/OpenPCDet
[3] https://github.com/TRAILab/CaDDN
[4] https://github.com/open-mmlab/OpenPCDet

duction network to obtain $\hat{F}_I \in \mathbb{R}^{W_I \times H_I \times C'}$, where $C' = 64$ is the number of reduced feature channels. For each position in $\hat{F}_I$, we predict its depth in a classification manner. Specifically, the continuous depth range $[d_{min}, d_{max}]$ is subdivided into D discrete bins using linear-increasing discretization (LID) as:

$$d_i = d_{min} + \frac{d_{max} - d_{min}}{D(D+1)} \cdot i(i+1), i \in [0, D] \quad (1)$$

where d_i is the discrete depth value of the i-th depth bin, $d_{min} = 0m$, $d_{max} = 73.6m$ and $D = 150$. We use a depth distribution estimation head DeepLabV3 [3] after the output image features of the image backbone $F_I^{out} \in \mathbb{R}^{W_I \times H_I \times C_{out}}$ ($W_I = 240$, $H_I = 75$ and $C_{out} = 2048$) to predict pixel-wise depth distribution $D_{depth} \in \mathbb{R}^{W_I \times H_I \times D}$ for each location in $\hat{F}_I$. We then calculate the outer product of $\hat{F}_I \in \mathbb{R}^{W_I \times H_I \times C'}$ and $D_{depth} \in \mathbb{R}^{W_I \times H_I \times D}$ to construct an image frustum grid $G \in \mathbb{R}^{W_I \times H_I \times D \times C'}$, where the feature of each location in $\hat{F}_I$ is placed at each predicted depth bin and weighted by the predicted probabilities. We then use an interpolation operation to obtain the cuboid shaped voxels in LiDAR coordinate(vehicle coordinate). Specifically, we first construct the voxel volume $Voxel_{Image}$ for the image-based branch in LiDAR coordinates, where the range of the 3D space is the same as the LiDAR branch, *i.e.*, $[0, 73.6] \times [-36.8, 36.8] \times [-2.0, 4.0]$ meters for X,Y and Z axis, and voxel size is set to $(0.4, 0.4, 0.4)$ meters, so we get $Voxel_{Image} \in \mathbb{R}^{184 \times 184 \times 15}$. For each voxel in $Voxel_{Image}$, we project the center coordinate of this voxel (x,y,z) into the image coordinate space to get (u,v,d) according to the projection relationship, and apply a trilinear interpolation operation to obtain the features for this voxel, and we thus get the voxel features of the image-based branch $F_I^v \in \mathbb{R}^{184 \times 184 \times 15 \times 64}$

Map to BEV. The voxel features of $F_I^v \in \mathbb{R}^{184 \times 184 \times 15 \times 64}$ are collapsed to a 2D BEV feature map by stacking the features in height dimension to obtain $\tilde{F}_I^{bev} \in \mathbb{R}^{184 \times 184 \times 15*64}$, which then goes through a channel compression network to get the image BEV feature map $F_I^{bev} \in \mathbb{R}^{184 \times 184 \times 128}$.

We illustrate the BEV feature map generating process for the image-based method in the bottom row of Fig. 2. More details can be found in CaDDN [11].

BEV Features Enhancement. We stack three Self-Calibrated Blocks (SCB) [9] after $F_I^{bev} \in \mathbb{R}^{184 \times 184 \times 128}$ to enhance the BEV features as:

$$\hat{F}_I^{bev} = SCB(F_I^{bev'}) \quad (2)$$

where $\hat{F}_I^{bev} \in \mathbb{R}^{184 \times 184 \times 128}$ is the enhanced BEV feature map.

3. Global BEV Feature Map Merging

We use the above operations to get the individual BEV feature maps for each camera for both LiDAR modality and image modality, and the 3D range for each camera is set to $[0, 73.6] \times [-36.8, 36.8] \times [-2.0, 4.0]$ for Front Camera, $[0, 73.6] \times [0, 73.6] \times [-2.0, 4.0]$ for Front Left Camera, $[0, 73.6] \times [-73.6, 0] \times [-2.0, 4.0]$ for Front Right Camera, $[-36.8, 36.8] \times [0, 73.6] \times [-2.0, 4.0]$ for Side Left Camera, and $[-36.8, 36.8] \times [-73.6, 0] \times [-2.0, 4.0]$ for Side Right Camera. We then use a global BEV feature map merging module to merge the individual BEV feature maps in both LiDAR-based branch and image-based branch to produce the global BEV feature maps BEV_{LiDAR}^{Global} and BEV_{Image}^{Global} with the global 3D range $[-36.8, 73.6] \times [-73.6, 73.6] \times [-2.0, 4.0]$. Specifically, for each of the individual BEV feature map BEV_m^i where $m \in [LiDAR, Image]$ and $i \in [0, 1, 2, 3, 4]$ in correspondence to 5 cameras, we first pad it with zeros to the global BEV range, and then use the $MaxPooling$ operation to merge the 5 individual BEV feature maps to a global one $BEV_{LiDAR}^{Global} \in \mathbb{R}^{276 \times 368 \times 128}$ or $BEV_{Image}^{Global} \in \mathbb{R}^{276 \times 368 \times 128}$, see Fig. 3.

4. BEV Backbone& Detection Heads

For the BEV backbone and detection heads, we simply use the original one in CenterPoint [11], *i.e.*, some basic convolution blocks after BEV feature map, a classification head to predict heatmaps and a regression head to predict parameters like 3D dimensions and orientations. Please refer to CenterPoint [11] and the implementation with Det3D[5] or OpenPCDet[6] for detail.

5. Cross-Modality Knowledge Distillation

5.1. Feature-based Knowledge Distillation

We use feature-based distillation between the BEV feature maps in LiDAR-based branch F_L^{bev} and image-based branch $\hat{F}_I^{bev}$ to make the image BEV features learn from the LiDAR BEV features containing accurate depth information and geometry information. We use mean square error(MSE) to minimize the difference between $\hat{F}_I^{bev}$ and F_L^{bev} as the feature distillation loss:

$$\mathcal{L}_{feat} = \mathcal{L}_{MSE}(\hat{F}_I^{bev}, F_L^{bev}) \quad (3)$$

5.2. Response-based Knowledge Distillation

To better leverage the useful information extracted by the teacher model, we further use the knowledge distillation in output level as the response-based distillation. Specifically, the predictions from the pre-trained teacher model are composed of two components, *i.e.*, the predicted heatmap

[5]https://github.com/tianweiy/CenterPoint
[6]https://github.com/open-mmlab/OpenPCDet

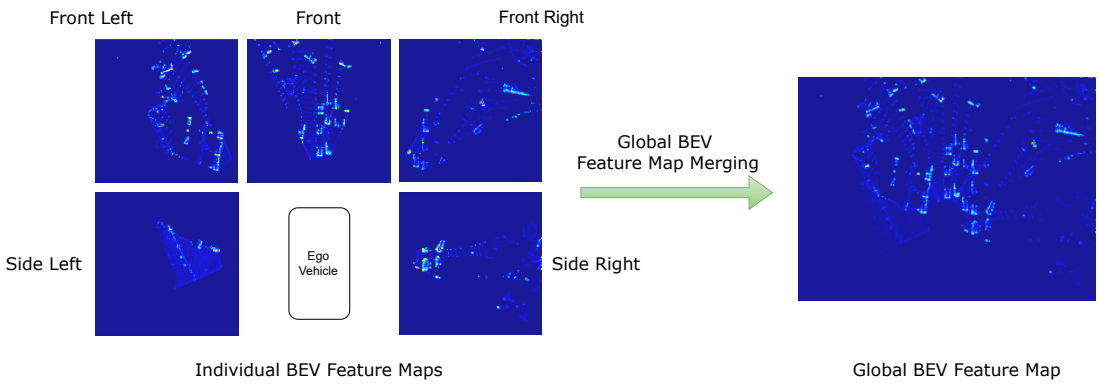

Figure 3. Global BEV Feature Map Merging.

$H^{teacher} \in \mathbb{R}^{276\times368\times3}$ where 3 corresponds to the 3 object classes used in this challenge, and the predicted regression values $R^{teacher} \in \mathbb{R}^{276\times368\times8}$, where 8 corresponds to $(\Delta x, \Delta y, z, l, w, h, sin\theta, cos\theta)$, and Δx, Δy are the center offsets, z is the z-component of the object center, l, w, h are the dimensions of the 3D box and θ is the rotation angle.

The detection losses include the classification loss $\mathcal{L}_{cls}$ and the regression loss $\mathcal{L}_{reg}$, and each loss includes two parts: loss with the **hard labels** and loss with the **soft labels**, *i.e.*, the predictions of the teacher model:

$$\mathcal{L}_{det} = \mathcal{L}_{cls} + \mathcal{L}_{reg} \tag{4}$$
$$\mathcal{L}_{cls} = \mathcal{L}_{cls}^{hard} + \mathcal{L}_{cls}^{soft} \tag{5}$$
$$\mathcal{L}_{reg} = \mathcal{L}_{reg}^{hard} + \mathcal{L}_{reg}^{soft} \tag{6}$$

Specifically, we use a weighted Focal Loss(FL) [8] to calculate the classification loss with hard labels.

$$\mathcal{L}_{cls}^{hard} = \frac{1}{N}\sum FL(h_i^{pred}, h_i^{gt}) \tag{7}$$

where $h_i^{pred} \in H^{pred}$ and $h_i^{gt} \in H^{gt}$, H^{pred} is the predicted heatmap of the image-based model, and H^{gt} is the heatmap generated by the ground-truth labels. N is the number of positive samples($h_i^{gt} = 1$) in H^{gt}. The element-wise loss is formulated as:

$$FL(y, y^{gt}) = -W\,(1-y')^{\gamma}\log(y') \tag{8}$$
$$y' = \begin{cases} y, \text{ when } y^{gt} = 1 \\ 1-y, \text{ when } y^{gt} < 1 \end{cases} \tag{9}$$
$$W = \begin{cases} 1, \text{ when } y^{gt} = 1 \\ (1-y^{gt})^{\beta}, \text{ when } y^{gt} < 1 \end{cases} \tag{10}$$

where $\gamma = 2$ and $\beta = 4$. We use Quality Focal Loss(QFL) [7] to calculate the classification loss with soft labels as:

$$\mathcal{L}_{cls}^{soft} = \frac{1}{N}\sum QFL(h_i^{pred}, h_i^{teacher}) \tag{11}$$

where $h_i^{pred} \in H^{pred}$ and $h_i^{gt} \in H^{teacher}$, H^{pred} is the predicted heatmap of the image-based model, and $H^{teacher}$ is the predicted heatmap of the teacher model. N is the number of positive samples($h_i^{teacher} > 0.3$) in $H^{teacher}$. The element-wise loss is formulated as:

$$QFL(y, y^{gt}) = -\left|y - y^{gt}\right|^{\gamma} CE(y, y^{gt}) \tag{12}$$
$$CE(y, y^{gt}) = ((1-y^{gt})\log(1-y) + y^{gt}\log y) \tag{13}$$

where $\gamma = 2$. Similarly, we use $SmoothL1$ loss to calculate the regression loss which contains two parts:

$$\mathcal{L}_{reg}^{hard} = SmoothL1(R^{pred}, R^{gt}) \tag{14}$$
$$\mathcal{L}_{reg}^{soft} = SmoothL1(R^{pred}, R^{teacher}) \tag{15}$$

where R^{pred} is the predicted regression values of the image-based model, R^{gt} is the ground-truth regression targets generated by hard labels and $R^{teacher}$ is the predicted regression values of the LiDAR-based teacher model.

6. Loss Function

The overall loss function is the combination of the losses mentioned above:

$$\begin{aligned}\mathcal{L}_{total} =& \lambda_1\mathcal{L}_{feat} + \lambda_2\mathcal{L}_{cls}^{hard} + \lambda_3\mathcal{L}_{cls}^{soft} \\ &+ \lambda_4\mathcal{L}_{reg}^{hard} + \lambda_5\mathcal{L}_{reg}^{soft}\end{aligned} \tag{16}$$

where $\lambda_1 - \lambda_5$ are hyper-parameters set to normalize the losses into similar scale, specifically, $\lambda_1 = 16$, $\lambda_2 = 1$, $\lambda_3 = 1$, $\lambda_4 = 1$ and $\lambda_5 = 4$.

7. Implementation Detail

7.1. Dataset and Metric

Experiments are conducted using the official data provided by Waymo Open Dataset (WOD) [2] for the 2022 Waymo 3D Camera-only Detection Challenge [1], which

consists of 798 training sequences, 202 validation sequences and 80 testing sequences. The main metric is the newly proposed Longitudinal Error Tolerant 3D Average Precision (LET-3D-AP) [6] which allows longitudinal localization errors of the predicted bounding boxes up to a given tolerance, and please refer to [6] for more details.

7.2. Training Strategy.

Teacher Model. We train the teacher model CenterPoint [11] with the official settings implemented by OpenPCDet[7]. The point cloud range is set to $[-25, 50] \times [-50, 50] \times [-2, 4]$ meters for the X,Y and Z axis. The voxel size is set to (0.05, 0.05, 0.1) meters. We train the teacher model with 2 NVIDIA 3090 GPUs with a total batch size of 16 for 20 epochs. we use AdamW optimizer with OneCycle learning rate strategy, the parameters are set as: max LR=2e-3, $\beta = (0.9, 0.999)$, eps=1e-8, weight decay=0.01. We sample 1 frame every 5 frames for training. Once the teacher model is pre-trained, we fix it during the training process of the student model.

Student Model. We train the model with 8 NVIDIA 3090 GPUs with a total batch size of 8, *i.e*. 1 batch (with 5 input images) per GPU. We first train the model with BEV feature distillation loss only for 5 epochs, which is similar to depth pre-training in a lot of monocular 3D detectors. We use AdamW optimizer with OneCycle learning rate strategy, the parameters are set as: max LR=2e-3, $\beta = (0.9, 0.999)$, eps=1e-8, weight decay=0.01. We then train the model with the complete losses for 10 epochs, we use AdamW optimizer with OneCycle learning rate strategy, the parameters are set as: max LR=3e-4, $\beta = (0.9, 0.999)$, eps=1e-8, weight decay=0.01. We sample 1 frame every 5 frames for training, *i.e*., we use 20% of the total training samples.

7.3. Data Augmentation.

We use no data augmentation.

7.4. Training&Testing Tricks.

We use no training&testing tricks.

8. Conclusion

In this work, We propose the Cross-Modality Knowledge Distillation (CMKD) Network for monocular 3D object detection. We take advantage of LiDAR data for useful knowledge transferring during training to boost the performance of the image-based method. Using only 20% of the total training samples, no previous frames, no data augmentation and no training&testing tricks, CMKD ranks among the top submissions on the leaderboard. And we believe the performance could be much further boosted with more engineering and fine-tuning. Additionally, CMKD can also be considered as a plug-and-play component for a BEV-based monocular 3D detector to further boost the performance without bringing any extra cost in the inference stage.

[7] https://github.com/open-mmlab/OpenPCDet

References

[1] 2022 waymo open dataset challenge: 3d camera-only detection. https://waymo.com/open/challenges/2022/3d-camera-only-detection/. 4

[2] Waymo open dataset. https://waymo.com/open/. 4

[3] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *CoRR*, abs/1706.05587, 2017. 3

[4] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Benjamin Sapp, Charles R. Qi, Yin Zhou, Zoey Yang, Aurelien Chouard, Pei Sun, Jiquan Ngiam, Vijay Vasudevan, Alexander McCauley, Jonathon Shlens, and Dragomir Anguelov. Large scale interactive motion forecasting for autonomous driving : The waymo open motion dataset. *CoRR*, abs/2104.10133, 2021. 2

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. IEEE Computer Society, 2016. 2

[6] Wei-Chih Hung, Henrik Kretzschmar, Vincent Casser, Jyh-Jing Hwang, and Dragomir Anguelov. Let-3d-ap: Longitudinal error tolerant 3d average precision for camera-only 3d detection. *arXiv preprint arXiv:2206.07705*, 2022. 5

[7] Xiang Li, Wenhai Wang, Lijun Wu, Shuo Chen, Xiaolin Hu, Jun Li, Jinhui Tang, and Jian Yang. Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems (NIPS)*, 2020. 4

[8] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 4

[9] Jiangjiang Liu, Qibin Hou, Ming-Ming Cheng, Changhu Wang, and Jiashi Feng. Improving convolutional networks with self-calibrated convolutions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10093–10102. Computer Vision Foundation / IEEE, 2020. 3

[10] Cody Reading, Ali Harakeh, Julia Chae, and Steven L. Waslander. Categorical depth distribution network for monocular 3d object detection. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[11] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking. *CVPR*, 2021. 2, 3, 5