Learning Distilled Collaboration Graph for Multi-Agent Perception

Anonymous Author(s)
Affiliation
Address
email

Abstract

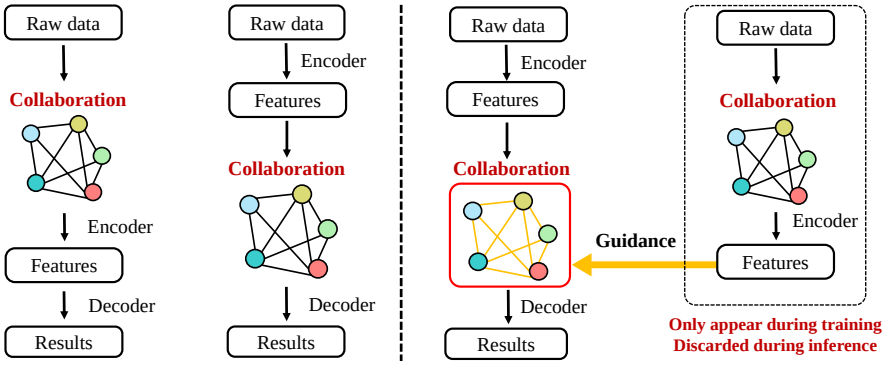
To promote better performance-bandwidth trade-off for multi-agent perception, we propose a novel *distilled collaboration graph* (DiscoGraph) to model trainable, pose-aware, and adaptive collaboration among agents. Our key novelties lie in two aspects. First, we propose a teacher-student framework to train DiscoGraph via knowledge distillation. The teacher model employs an early collaboration with holistic-view inputs; the student model is based on intermediate collaboration with single-view inputs. Our framework trains DiscoGraph by constraining post-collaboration feature maps in the student model to match the correspondences in the teacher model. Second, we propose a matrix-valued edge weight in DiscoGraph. In such a matrix, each element reflects the inter-agent attention at a specific spatial region, allowing an agent to adaptively highlight the informative regions. During inference, we only need to use the student model, which we call the distilled collaboration network (DiscoNet). Attributed to the teacher-student framework, multiple agents with the shared DiscoNet could collaboratively approach the performance of a hypothetical teacher model with a holistic view. Our approach is validated on a large-scale multi-agent 3D object detection dataset that we synthesized using CARLA and SUMO co-simulation. Our experiments show that the proposed DiscoNet could achieve 192 times less communication volume and still outperforms the state-of-the-art methods for both AP@0.5 and 0.7.

1 Introduction

Perception, which involves organizing, identifying and interpreting sensory information, is a crucial ability for intelligent agents to understand the environment. Single-agent perception [4] has been studied extensively in recent years, *e.g.*, 2D/3D object detection [16, 24], tracking [20, 19] and segmentation [22, 15], *etc.* Despite its great progress, single-agent perception suffers from a number of shortcomings stemmed from its individual perspective. For example, in autonomous driving [8], the LiDAR-based perception system can hardly perceive the target in the occluded or long-range areas. Intuitively, with an appropriate collaboration strategy, multi-agent perception could fundamentally upgrade the perception ability over single-agent perception.

To design a collaboration strategy, current approaches mainly include raw-measurement-based early collaboration, output-based late collaboration and feature-based intermediate collaboration. Early collaboration [3] aggregates the raw measurements from all the agents, promoting a holistic perspective; see Fig. 1 (a). It can fundamentally solve the occlusion and long-range issues occurring in the single-agent perception; however, it requires a lot of communication bandwidth. Contrarily, late collaboration aggregates each agent's perception outputs. Although it is bandwidth-efficient, each individual perception output could be noisy and incomplete, causing unsatisfying fusion results. To deal with the performance-bandwidth trade-off, intermediate collaboration [17, 31, 18] has been proposed to aggregate intermediate features across agents; see Fig. 1 (b). Since we can

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.



(a) Early (b) Intermediate (c) Ours

Figure 1: Scheme comparison. (a) Early collaboration requires an expensive bandwidth for raw data transmission. (b) Intermediate collaboration needs an appropriate collaboration strategy. (c) The proposed method incorporates both early and intermediate collaboration into a knowledge distillation framework, enabling the knowledge of early collaboration to guide the training of an intermediate collaboration strategy, leading to better trade-off between performance and bandwidth.

squeeze representative information to compact features, this approach can potentially both achieve communication bandwidth efficiency and upgrade perception ability; however, a bad design of collaboration strategy might cause information loss during feature abstraction and fusion, causing limited improvement of the perception ability.

To achieve an effective design of intermediate collaboration, we propose a distilled collaboration graph (DiscoGraph) to model the collaboration among agents. In DiscoGraph, each node is an agent with real-time pose information and each edge reflects the pair-wise collaboration between two agents. The proposed DiscoGraph is trainable, pose-aware, and adaptive to real-time measurements, reflecting dynamic collaboration among agents. It is novel from two aspects. First, from the training aspect, we propose a teacher-student framework to train DiscoGraph through knowledge distillation [1, 10, 23]; see Fig. 1 (c). Here the teacher model is based on early collaboration with holistic-view inputs and the student model is based on intermediate collaboration with single-view inputs. The knowledge-distillation-based framework enhances the training of DiscoGraph by constraining the post-collaboration feature maps in the student model to match the correspondences in the teacher model. With the guidance of both output-level supervision from perception and feature-level supervision from knowledge distillation, the distilled collaboration graph promotes better feature abstraction and aggregation, improving the performance-bandwidth trade-off. Second, from the modeling aspect, we propose a matrix-valued edge weight in DiscoGraph to reflect the collaboration strength with a high spatial resolution. In the matrix, each element represents the inter-agent attention at a specific spatial region. This design allows the agents to adaptively highlight the informative regions and strategically select appropriate partners to request supplementary information.

During inference, we only need to use the student model. Since it leverages DiscoGraph as the key component, we call the student model as the distilled collaboration network (DiscoNet). Multiple agents with the shared DiscoNet could collaboratively approach the performance of a hypothetical teacher model with the holistic view.

To validate the proposed method, we build a new large-scale multi-agent 3D object detection dataset in autonomous driving scenarios based on CARLA and SUMO co-simulation platform [6]. Comprehensive experiments conducted in 3D detection [33, 25, 26] have shown that the proposed DiscoNet achieves better performance-bandwidth trade-off than state-of-the-art intermediate collaboration methods. For example, DiscoNet could achieve 192 times less communication volume and still outperforms V2VNet [31] for both AP@0.5 and 0.7. Our code and dataset will be released.

2 Multi-Agent Perception System with Distilled Collaboration Graph

This work considers a collaborative perception system to perceive a scene, where multiple agents can perceive and collaborate with each other through a broadcast communication channel. We assume each agent is provided with an accurate pose and the perceived measurements are well synchronized. Then, given a certain communication bandwith, we aim to maximize the perception ability of

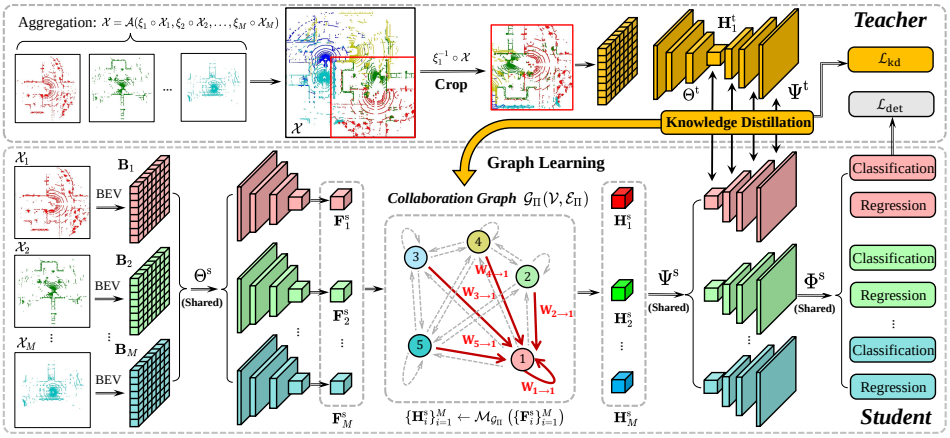


Figure 2: Overall teacher-student training framework. In the student model, for Agent 1 (in red), its input single-view point cloud can be converted to a BEV map, and then be consumed by a shared encoder Θ^{s} to obtain the feature map $\mathbf{F}_i^{\mathrm{s}}$. Based on the collaboration graph $\mathcal{G}_\Pi$, Agent 1 aggregates the neural messages from other agents and obtains the updated feature map $\mathbf{H}_i^{\mathrm{s}}$. The shared header following the shared decoder outputs the 3D detection results. In the teacher model, we aggregate the points collected from all agents to obtain a holistic-view point cloud. We crop the region and align the pose to ensure the BEV maps in both teacher and student models are synchronized. We constrain all the post-collaboration feature maps in the student model to match the correspondences in the teacher model through knowledge distillation, resulting in a collaborative student model.

each agent through optimizing a collaboration strategy. To design such a strategy, we consider a teacher-student training framework to integrate the strengths from both early and intermediate collaboration. During training, we leverage an early-collaboration model (teacher) to teach an intermediate-collaboration model (student) how to collaborate through knowledge distillation. Here the teacher model and the student model are shared by all the agents, but each agent would input raw measurements from its own view to either model; see Fig. 2. During inference, we only need the student model, where multiple agents with the shared student model could collaboratively approach the performance of the teacher model with the hypothetical holistic view.

In this work, we focus on the perception task of LiDAR-based 3D object detection while the proposed methods can be broadly applied to many other perception tasks.

2.1 Student: Intermediate collaboration via graphs

Feature encoder. The functionality is to extract informative features from raw measurements for each agent. Let $\mathcal{X}_i$ be the 3D point cloud collected by the ith agent, the feature of the ith agent is obtained as $\mathbf{F}_i^{\mathrm{s}} \leftarrow \Theta^{\mathrm{s}}(\mathcal{X}_i)$, where $\Theta^{\mathrm{s}}(\cdot)$ is the feature encoder shared by all the agents and the superscript s reflects the student mode. To implement the encoder, we convert a 3D point cloud to a bird's-eye-view (BEV) map, which is amenable to classic 2D convolutions. Specifically, we quantize the 3D points into regular voxels and represent the 3D voxel lattice as a 2D pseudo-image, with the height dimension corresponding to image channels. Such a 2D image is virtually a BEV map, whose basic element is a cell that is associated with a binary vector along the vertical axis. Let $\mathbf{B}_i \in \{0,1\}^{K\times K\times C}$ be the BEV map of the ith agent associated with $\mathcal{X}_i$. With this map, we can apply four blocks composed of 2D convolutions, batch normalization and ReLU activation to gradually reduce the spatial dimension and increase the number of channels for the BEV, to obtain the feature map $\mathbf{F}_i^{\mathrm{s}} \in \mathbb{R}^{\bar{K}\times\bar{K}\times\bar{C}}$ with $\bar{K}\times\bar{K}$ the spatial resolution and $\bar{C}$ the number of channels.

Feature compression. To save the communication bandwidth, each agent could compress its feature map prior to transmission. Here we consider a 1×1 convolutional autoencoder [21] to compress/decompress the feature maps along the channel dimension. The autoencoder is trained together with the whole system, making the system work with limited collaboration information.

Collaboration graph process. The functionality is to update the feature map through data transmission among the agents. The core component here is a *collaboration graph* $\mathcal{G}_\Pi(\mathcal{V}, \mathcal{E}_\Pi)$, where $\mathcal{V}$ is the fixed node set and $\mathcal{E}_\Pi$ is the trainable edge set. Each node in $\mathcal{V}$ is an agent with the real-time pose information; for instance, $\boldsymbol{\xi}_i \in \mathfrak{se}(3)$ is the ith agent's pose in the global coordinate system; and each

edge in $\mathcal{E}_\Pi$ is trainable and models the collaboration between two agents, with Π an edge-weight encoder, reflecting the trainable collaboration strength between agents. Let $\mathcal{M}_{\mathcal{G}_\Pi}(\cdot)$ be the collaboration process defined on the collaboration graph $\mathcal{G}_\Pi$. The feature maps of all the agents after collaboration are $\{\mathbf{H}_i^{\mathrm{s}}\}_{i=1}^M \leftarrow \mathcal{M}_{\mathcal{G}_\Pi}\left(\{\mathbf{F}_i^{\mathrm{s}}\}_{i=1}^M\right)$. This process has three stages: neural message transmission (**S1**), neural message attention (**S2**) and neural message aggregation (**S3**).

In the *neural message transmission stage* (**S1**), each agent transmits its BEV-based feature map to the other agents. Since the BEV-based feature map summarizes the information of each agent, we consider it as a neural message. In the *neural message attention stage* (**S2**), each agent receives others' neural messages and determines the matrix-valued edge weights, which reflect the importance of the neural message from one agent to another at each individual cell. Since each agent has its unique pose, we leverage the collaboration graph to achieve feature transformation across agents. For the ith agent, the transformed BEV-based feature map from the jth agent is then $\mathbf{F}_{j\to i}^{\mathrm{s}} = \Gamma_{j\to i}(\mathbf{F}_i^{\mathrm{s}}) \in \mathbb{R}^{\bar{K}\times\bar{K}\times\bar{C}}$, where the transformation $\Gamma_{j\to i}(\cdot)$ is based on two ego poses $\boldsymbol{\xi}_j$ and $\boldsymbol{\xi}_i$. Now $\mathbf{F}_{j\to i}^{\mathrm{s}}$ and $\mathbf{F}_i^{\mathrm{s}}$ are supported in the same coordinate system. To determine the edge weights, we use the edge encoder Π to correlate the ego feature map and the feature map from another agent; that is, the matrix-valued edge weight from the jth agent to the ith agent is $\mathbf{W}_{j\to i} = \Pi(\mathbf{F}_{j\to i}^{\mathrm{s}}, \mathbf{F}_i^{\mathrm{s}}) \in \mathbb{R}^{\bar{K}\times\bar{K}}$, where Π concatenates two feature maps along the channel dimension and then uses four 1×1 convolutional layers to gradually reduce the number of channels from $2\bar{C}$ to 1. Note that previous works [17, 18, 31] generally consider a scalar-valued edge weight to reflect the overall collaboration strength between two agents; while we consider a matrix-valued edge weight $\mathbf{W}_{j\to i}$, which models the collaboration strength from the jth agent to the ith agent with a $\bar{K}\times\bar{K}$ spatial resolution. In this matrix, each element corresponds to a cell in the BEV map, indicating a specific spatial region; thus, this matrix reflects the spatial attention at a cell-level resolution. In the *neural message aggregation stage* (**S3**), each agent aggregates the feature maps from all the agents based on the corresponding matrix-valued edge weights. The updated feature map of the agent i is $\mathbf{H}_i^{\mathrm{s}} = \sum_{j=1}^M \mathbf{W}_{j\to i} \odot \mathbf{F}_{j\to i}^{\mathrm{s}}$, where $\odot$ denotes the dot product with channel-wise broadcasting.

Remark. The proposed collaboration graph is trainable because each matrix-valued edge weight is a trainable matrix to reflect the agent-to-agent attention in a cell-level spatial resolution; it is pose-aware, empowering all the agents to work with the synchronized coordinate system; furthermore, it is dynamic at each timestamp as each edge weight would adapt to the real-time neural messages. According to the proposed collaboration graph, the agents can discover the region requiring collaboration on the fly, and strategically select appropriate partners to request supplementary information.

Decoder and header. After collaboration, each agent decodes the updated BEV-based feature map. The decoded feature map is $\mathbf{M}_i^{\mathrm{s}} \leftarrow \Psi^{\mathrm{s}}(\mathbf{H}_i^{\mathrm{s}})$. To implement the decoder $\Psi^{\mathrm{s}}(\cdot)$, we progressively up-sample $\mathbf{H}_i^{\mathrm{s}}$ with four layers, where each layer first concatenates the previous feature map with the corresponding feature map in the encoder and then uses a 1×1 convolutional operation to halve the number of channels. Finally, we use an output header to generate the final detection outputs, $\widehat{\mathbf{Y}}_i^{\mathrm{s}} \leftarrow \Phi^{\mathrm{s}}(\mathbf{M}_i^{\mathrm{s}})$. To implement the header $\Psi^{\mathrm{s}}(\cdot)$, we use two branches of convolutional layers to classify the foreground-background categories and regress the bounding boxes.

2.2 Teacher: Early collaboration

During the training phase, an early-collaboration model, as the teacher, is introduced to guide the intermediate-collaboration model, which is a student. Similar to the student model, the teacher's pipeline has the feature encoder Θ^{t}, feature decoder Ψ^{t} and the output header Φ^{t}. Note that all the agents share the same teacher model to guide one student model; however, each agent provides the inputs with its own pose, and its inputs to both the teacher and student models should be well aligned.

Feature encoder. Let $\mathcal{X} = \mathcal{A}(\boldsymbol{\xi}_1 \circ \mathcal{X}_1, \boldsymbol{\xi}_2 \circ \mathcal{X}_2, ..., \boldsymbol{\xi}_M \circ \mathcal{X}_M)$ be a holistic-view 3D point cloud that aggregates all the points from all M agents in the global coordinate system, where $\mathcal{A}(\cdot, ..., \cdot)$ is the aggregation operator of multiple 3D point clouds, and $\boldsymbol{\xi}_i$ and $\mathcal{X}_i$ are the pose and the 3D point cloud of the ith agent, respectively. To ensure the inputs to the teacher model and the student model are aligned, we transform the holistic-view point cloud $\mathcal{X}$ to an agent's own coordinate based on the pose information. Now, for the ith agent, the input to the teacher model $\boldsymbol{\xi}_i^{-1} \circ \mathcal{X}$ and the input to the student model $\mathcal{X}_i$ are in the same coordinate system. Similarly to the feature encoder in the student model, we convert the 3D point cloud $\boldsymbol{\xi}_i^{-1} \circ \mathcal{X}$ to a BEV map and use 2D convolutions to obtain the feature map of the ith agent in the teacher model, $\mathbf{H}_i^{\mathrm{t}} \in \mathbb{R}^{\bar{K}\times\bar{K}\times\bar{C}}$. Here we crop the BEV map to ensure it has the same spatial range and resolution with the BEV map in the student model.

Decoder and header. Similarly to the decoder and header in the student model, we adopt $\mathbf{M}_i^{\mathrm{t}} \leftarrow \Psi^{\mathrm{t}}(\mathbf{H}_i^{\mathrm{t}})$ to obtain the decoded BEV-based feature map and $\widehat{\mathbf{Y}}_i^{\mathrm{t}} \leftarrow \Phi^{\mathrm{t}}(\mathbf{M}_i^{\mathrm{t}})$ to obtain the predicted foreground-background categories and regressed bounding boxes.

Teacher training scheme. As in common teacher-student frameworks, we train the teacher model separately. We employ the binary cross-entropy loss to supervise foreground-background classification and the smooth L_1 loss to supervise the bounding-box regression. Overall, we minimize the loss function $\mathcal{L}^{\mathrm{t}} = \sum_{i=1}^{M} \mathcal{L}_{\mathrm{det}}(\mathbf{Y}_i^{\mathrm{t}}, \widehat{\mathbf{Y}}_i^{\mathrm{t}})$, where classification and regression losses are collectively denoted as $\mathcal{L}_{\mathrm{det}}$ and $\mathbf{Y}_i^{\mathrm{t}} = \mathbf{Y}_i^{\mathrm{s}}$ is the ground-truth detection in the perception region of the ith agent.

2.3 System training with knowledge distillation

Given a well-trained teacher model, we use both detection loss and knowledge distillation loss to supervise the training of the student model. We consider minimizing the following loss

$$\mathcal{L}^{\mathrm{s}} = \sum_{i=1}^{M} \left(\mathcal{L}_{\mathrm{det}}(\mathbf{Y}_i^{\mathrm{s}}, \widehat{\mathbf{Y}}_i^{\mathrm{s}}) + \lambda_{\mathrm{kd}} \mathcal{L}_{\mathrm{kd}}(\mathbf{H}_i^{\mathrm{s}}, \mathbf{H}_i^{\mathrm{t}}) + \lambda_{\mathrm{kd}} \mathcal{L}_{\mathrm{kd}}(\mathbf{M}_i^{\mathrm{s}}, \mathbf{M}_i^{\mathrm{t}}) \right).$$

The detection loss $\mathcal{L}_{\mathrm{det}}$ is similar to that of the teacher, including both foreground-background classification loss and the bounding box regression loss, pushing the detection result of each agent to be close to its local ground-truth. The second and third terms form a knowledge distillation loss, regularizing the student model to generate similar feature maps with the teacher model. The hyperparameter λ_{kd} controls the weight of the knowledge distillation loss $\mathcal{L}_{\mathrm{kd}}$ defined as follows

$$\mathcal{L}_{\mathrm{kd}}(\mathbf{H}_i^{\mathrm{s}}, \mathbf{H}_i^{\mathrm{t}}) = \sum_{n=1}^{\bar{K} \times \bar{K}} D_{\mathrm{KL}} \left(\sigma\left((\mathbf{H}_i^{\mathrm{s}})_n \right) || \sigma\left(\left(\mathbf{H}_i^{\mathrm{t}} \right)_n \right) \right) \right),$$

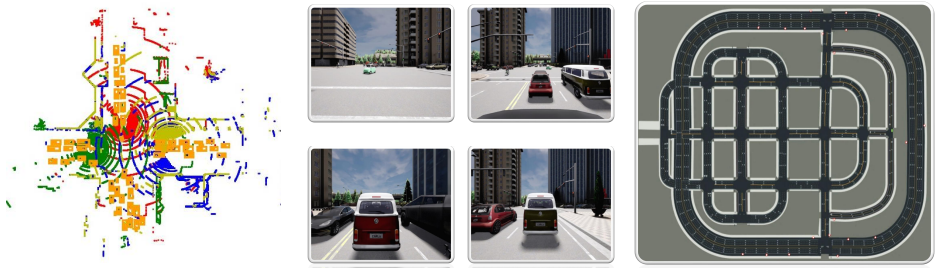
where $D_{KL}(p(\mathbf{x})||q(\mathbf{x}))$ denotes the Kullback-Leibler (KL) divergence of distribution $q(\mathbf{x})$ from distribution $p(\mathbf{x})$, $\sigma(\cdot)$ indicates the softmax operation of the feature vector along the channel dimension, and $(\mathbf{H}_i^{\mathrm{s}})_n$ and $(\mathbf{H}_i^{\mathrm{t}})_n$ denote the feature vectors at the nth cell of the ith agent's feature map in the student model and teacher model, respectively. Similarly, the loss on decoded feature maps $\mathcal{L}_{\mathrm{kd}}(\mathbf{M}_i^{\mathrm{s}}, \mathbf{M}_i^{\mathrm{t}})$ can be introduced to enhance the regularization.

Remark. As mentioned in Section 2.1, the feature maps output by the collaboration graph process in the student model is computed by $\{\mathbf{H}_i^{\mathrm{s}}\}_{i=1}^{M} \leftarrow \mathcal{M}_{\mathcal{G}_\Pi}\left(\{\mathbf{F}_i^{\mathrm{s}}\}_{i=1}^{M}\right)$. Intuitively, the feature map of the teacher model $\{\mathbf{H}_i^{\mathrm{t}}\}_{i=1}^{M}$ would be the desired output of the collaboration graph process $\mathcal{M}_{\mathcal{G}_\Pi}(\cdot)$. Therefore, we constrain all the post-collaboration feature maps in the student model to match the correspondences in the teacher model through knowledge distillation. This constraint would further regularize the upfront trainable components: i) the distilled student encoder Θ^{s}, which abstracts the features from raw measurements and produces the input to $\mathcal{M}_{\mathcal{G}_\Pi}(\cdot)$, and ii) the edge-weight encoder Π in the distilled collaboration graph. Consequently, through knowledge distillation and back-propagation, the distilled student encoder would learn to abstract informative features from raw data for better collaboration; and the distilled edge-weight encoder would learn how to control the collaboration based on agents' features. In a word, our distilled collaboration network (DiscoNet) can comprehend feature abstraction and fusion via the proposed knowledge distillation framework.

3 Related Work

Multi-agent communication. As core components in a multi-agent system, communication strategy among agents has been actively studied in previous works [27, 7, 28]. For example, CommNet [29] adopted the averaging operation to achieve continuous communication in multi-agent system; VAIN [11] considered an attention mechanism to determine which agents would share information; ATOC [12] exploited an attention unit to determine what information to share with other agents; and TarMAC [5] implicitly learned a signature-based soft attention mechanism to let agents actively select which agents should receive the messages. In this work, we focus on the 3D perception task and propose a trainable, dynamic collaboration graph to control the communication among agents.

Collaborative perception. Collaborative perception is an application of multi-agent communication system to perception tasks [9, 32, 13]. Who2com [18] exploited a handshake communication mechanism to determine which two agents should communicate for image segmentation; When2com [17] introduced an asymmetric attention mechanism to decide when to communicate and how to create communication groups for image segmentation; V2VNet [31] proposed multiple rounds of message passing on a spatial-aware graph neural network for joint perception and prediction in autonomous



(a) Point clouds (b) RGB images (c) The map of *Town05*

Figure 3: Created multi-agent perception dataset. (a) Point clouds from multiple agents in a top-down view (each color indicates an individual agent). (b) RGB images captured by four agents. (c) The map of *Town05* which is a squared-grid town with cross junctions and multiple lanes per direction.

Table 1: Detection comparison. Our method is in bold and * indicates the pose-aware version. The proposed DiscoNet is the best among intermediate collaboration. Even with 16 times compression, DiscoNet(16) still outperforms V2VNet without compression.

Method	Collaboration Approach			Average Precision (AP)	
	Early	Intermediate	Late	IoU=0.5	IoU=0.7
Upper-bound	✓	✗	✗	64.2	60.3
	✓	✗	✓	60.0	47.6
When2com* [17]	✗	✓	✗	47.9	42.9
When2com [17]	✗	✓	✗	46.7	42.4
Who2com* [18]	✗	✓	✗	47.2	42.2
Who2com [18]	✗	✓	✗	46.3	42.0
V2VNet [31]	✗	✓	✗	57.0	49.1
DiscoNet	✗	✓	✗	**60.2**	**54.0**
DiscoNet(16)	✗	✓	✗	**59.0**	**53.8**
Lower-bound	✗	✗	✓	55.4	41.8
	✗	✗	✗	45.8	40.6

Table 2: Detection performance by regularizing various layers via knowledge distillation (KD). $\mathbf{M}_i^s\{n\}(n = 1, 2, 3, 4)$ denotes different layers in the decoder. The performance significantly boots once the KD regularization is applied. Further regularization has a slight effect on the detection performance.

KD Regularization					Average Precision (AP)	
$\mathbf{H}_i^s$	$\mathbf{M}_i^s\{1\}$	$\mathbf{M}_i^s\{2\}$	$\mathbf{M}_i^s\{3\}$	$\mathbf{M}_i^s\{4\}$	IoU=0.5	IoU=0.7
✗	✗	✗	✗	✗	56.2	50.8
✓	✗	✗	✗	✗	59.0	53.4
✓	✓	✗	✗	✗	58.4	52.3
✓	✓	✓	✗	✗	59.0	52.8
✓	✓	✓	✓	✗	**60.2**	**54.0**
✓	✓	✓	✓	✓	60.0	53.5
✗	✓	✓	✓	✓	58.7	53.3

driving; and [30] proposed a pose error regression module to learn to correct pose errors when the pose information from other agents is noisy.

Advantages and limitation of DiscoNet. First, previous collaborative perception only rely on the final detection supervision; while DiscoNet leverages both detection supervision and intermediate feature supervision, acquiring more explicit guidance. Second, the collaboration attention in previous methods is a scalar, which cannot reflect the significance of each region; while DiscoNet uses a matrix, leading to more flexible collaboration at various spatial regions. Third, previous methods uses multiple-round collaboration: When2com needs at least one more round of communication after the first handshake, and V2VNet claims three rounds to ensure reliable performance; while DiscoNet only requires one round, suffering less from latency. Admittedly, DiscoNet has a limitation by assuming accurate pose for each agent, which could be improved by method like [30].

4 Experiment

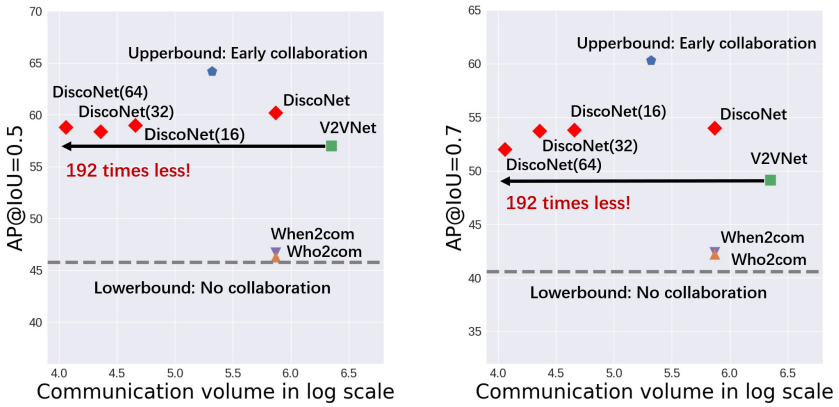
4.1 Multi-agent 3D object detection dataset

Since there is no public dataset to support the research on multi-agent 3D object detection, we build such a dataset based on both SUMO [14], a micro-traffic simulation, and CARLA [6], a widely-used open-source simulator for autonomous driving research. SUMO is firstly used to produce numerically-realistic traffic flow and CARLA is employed to get realistic sensory streams including LiDAR and RGB data, from multiple vehicles located in the same geographical area. The simulated LiDAR is rotated at a frequency of 20Hz and has 32 channels; the range which denotes the maximum distance to capture is 70 meters, and the number of points per second is set as 250,000. The RGB camera images are also stored and can be used in multi-agent multi-modality perception in the future. We use *Town05* with multiple lanes and cross junctions for dense traffic simulation; see Fig. 3.

The proposed dataset follows the same storage format of nuScenes [2], a widely-used large-scale single-agent autonomous driving dataset. nuScenes collected real-world single-agent data; while we

Table 3: Comparison of various intermediate collaboration strategies. KD indicates knowledge distillation. With KD and matrix-valued edge weights, our DiscoGraph outperforms the others.

Collaboration Strategy	Scalar Weight				Matrix Weight		
	No	Sum	Average	Weighted Average	Max	Cat	DiscoGraph
w/o KD (AP@IoU 0.5/0.7)	45.8/40.6	54.1/49.5	56.1/51.5	55.2/49.7	**57.9/52.1**	54.5/49.0	56.2/50.8
w/ KD (AP@IoU 0.5/0.7)	45.6/41.9	56.0/51.3	57.0/51.9	56.0/50.6	57.4/51.8	56.3/51.3	**60.2/54.0**
Gain	-0.2/1.3	1.9/1.8	0.9/0.4	0.8/0.9	-0.5/-0.3	1.8/2.3	**4.0/3.2**
Gain (%)	-0.4/3.2	3.5/3.6	1.6/0.8	1.4/1.8	-0.9/-0.6	3.3/4.7	**7.1/6.3**



(a) Scatterplot in AP@IoU 0.5 (b) Scatterplot in AP@IoU 0.7

Figure 4: Performance-bandwidth trade-off. DiscoNet(64) achieves 192 times less communication volume and still outperforms V2VNet [31].

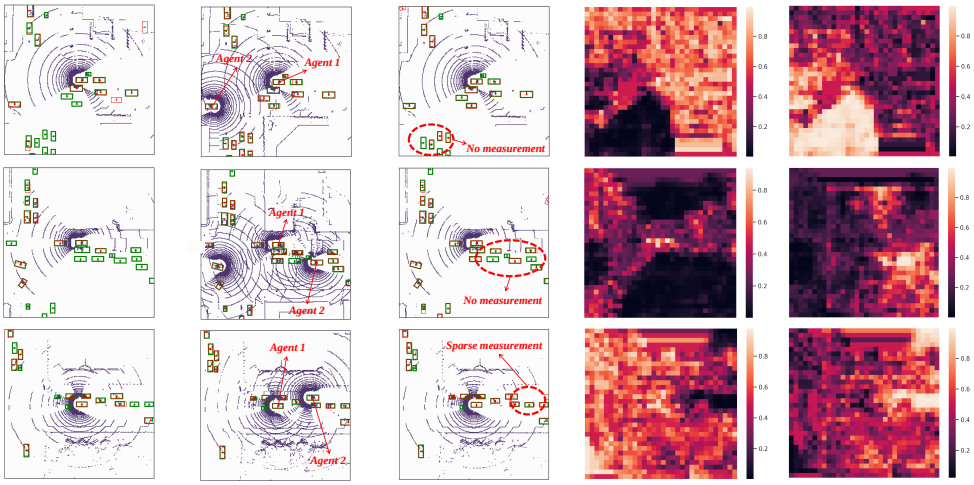
simulate multi-agent scenarios. Each scene includes a 20 second traffic flow at a certain intersection of *Town05*, and the LiDAR streams are recorded every 0.2 second, which means that each scene consists of 100 frames. We generate 100 scenes with a total of 10,000 frames, and in each scene, we randomly select 2-5 vehicles as the collaboration agents. We use 8,000/900/1,100 frames for training/validation/testing. Each frame has multiple samples, and there are 23,500 samples in the training set and 3,100 samples in the test set.

4.2 Quantitative evaluation

Implementation and evaluation. We crop the points located in the region of $[-32, 32] \times [-32, 32] \times [-3, 2]$ meters defined in the ego-vehicle XYZ coordinate. We set the width/length of each voxel as 0.25 meter, and the height as 0.4 meter; therefore the BEV map input to the student/teacher encoder has a dimension of $256 \times 256 \times 13$. The intermediate feature output by the encoder has a dimension of $32 \times 32 \times 256$, and we use an autoencoder (AE) for compression to save bandwidth: the sent message is the embedding output by AE. After the agents receive the embedding, they will decode it to the original resolution $32 \times 32 \times 256$ for intermediate collaboration. The hyperparameter λ_{kd} is set as 10^5. We train all the models using NVIDIA 3090 GPU with 200,000 iterations. The batch size is set as 4, and the learning rate is set as 0.001. We employ the generic BEV detection evaluation metric: Average Precision (AP) at Intersection-over-Union (IoU) threshold of 0.5 and 0.7. We target the detection of the car category and report the results on the test set.

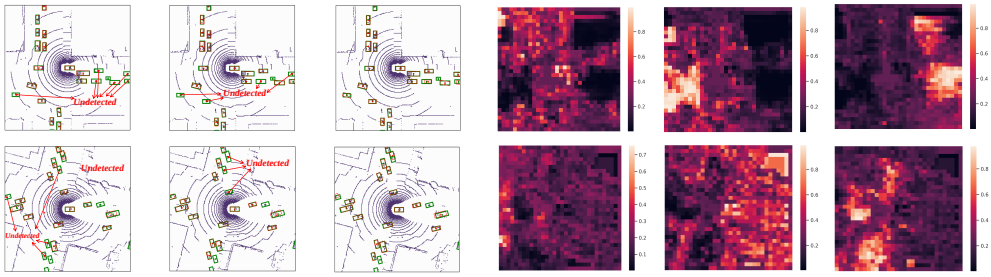
Baselines and existing methods. We consider the single-agent perception model; called *Lower-bound*, which consumes a single-view point cloud. The teacher model based on early collaboration with a holistic view is considered as the *Upper-bound*. Both lower-bound and upper-bound can involve late collaboration, which aggregates all the boxes and applies a global non-maximum suppression (NMS). We also consider three intermediate-collaboration methods, Who2com [18], When2com [17] and V2VNet [31]. Since the original Who2com and When2com do not consider pose information, we consider both pose-aware and pose-agnostic versions to achieve fair comparisons. All the methods use the same detection architecture and conduct collaboration at the same intermediate feature layer.

Results and comparisons. Table 1 shows the comparisons in terms of AP (@IoU = 0.5/0.7). We see that i) early collaboration (first row) achieves the best detection performance, and there is an obvious improvement over no collaboration (last row), *i.e.*, AP@0.5 and AP@0.7 are increased by 40.2% and



(a) Lower-bound (b) Upper-bound (c) DiscoNet (d) $\mathbf{W}_{1\to1}$ (e) $\mathbf{W}_{2\to1}$

Figure 5: Detection and matrix-valued edge weight for Agent 1. Green and red boxes denote ground-truth (GT) and predictions, respectively. (a-c) shows the outputs of lower-bound, upper-bound and DiscoNet compared to GT. (d) Ego matrix-valued edge weight for Agent 1. Attention to the region with sparse/no measurement is suppressed. (e) Matrix-valued edge weight from Agent 2 to Agent 1. Attention to the region with complementary information from Agent 2 is enhanced.



(a) When2com (b) V2VNet (c) DiscoNet (d) $\mathbf{W}_{1\to1}$ (e) $\mathbf{W}_{2\to1}$ (f) $\mathbf{W}_{3\to1}$

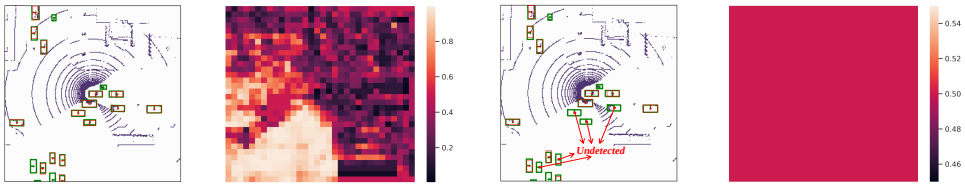
Figure 6: DiscoNet qualitatively outperforms the state-of-the-art methods. Green and red boxes denote ground-truth and detection, respectively. (a) Output of when2com. (b) Output of V2VNet. (c) Output of DiscoNet. (d)-(f) Matrix-valued edge weights. (Ego agent: 1; neighbour agents: 2 and 3.)

48.5% respectively, reflecting the significance of collaboration; ii) late collaboration improves the lower-bound (second last row), but hurts the upper-bound (second row), reflecting the unreliability of late collaboration. This is because the final, global NMS can remove a lot of noisy boxes for the lower-bound, but would also remove many useful boxes for the upper-bound; and iii) among the intermediate collaboration-based methods, the proposed DiscoNet achieves the best performance. Comparing to the pose-aware When2com, DiscoNet improves by 25.7% in AP@0.5 and 25.9% in AP@0.7. Comparing to V2VNet, DiscoNet improves by 5.6 % in AP@0.5 and 10.0 % in AP@0.7. Even with 16-times compression by the autoencoder, DiscoNet(16) still outperforms V2VNet.

Performance-bandwidth trade-off analysis. Fig. 4 thoroughly compares the proposed DiscoNet with the baseline methods in terms of the trade-off between detection performance and communication bandwidth. In Plots (a) and (b), the lower-bound with no collaboration is shown as a dashed line since the communication volume is zero, and the other method is shown as one dot in the plot. We see that i) DiscoNet with feature-map compression by an autoencoder achieves a far-more superior trade-off than the other methods; ii) DiscoNet(64), which is DiscoNet with 64-times feature compression by autoencoder, achieves 192 times less communication volume and still outperforms V2VNet for both AP@0.5 and 0.7; and iii) feature compression does not significantly hurt the detection performance.

4.3 Qualitative evaluation

Visualization of edge weight. To understand the working mechanism of the proposed collaboration graph, we visualize the detection results and the corresponding edge weights; see Fig. 5. Note that the proposed edge weight is a matrix, reflecting the collaboration attention in a cell-level resolution,



(a) Output w/ teacher (b) $\mathbf{W}_{2\to1}$ w/ teacher (c) Output w/o teacher (d) $\mathbf{W}_{2\to1}$ w/o teacher

Figure 7: Knowledge distillation from the teacher is critical to DiscoNet. Without knowledge distillation regularization, the matrix-valued edge weight does not reflect informative spatial attention.

which is shown as a heat map. Fig. 5 (a), (b) and (c) show three exemplar detection results of Agent 1 based on the lower-bound, upper-bound and the proposed DiscoNet, respectively. We see that with the collaboration, DiscoNet is able to detect boxes in those occluded and long-range regions. To understand why, Fig. 5 (d) and (e) provide the corresponding ego edge weight and the edge weight from agent 2 to agent 1, respectively. We clearly see that with the proposed collaboration graph, Agent 1 is able to receive complementary information from the other agents.

Comparison with When2com and V2VNet. Fig. 6 shows two examples to compare the proposed DiscoNet with When2com, V2VNet. We see that DiscoNet is able to detect more objects. The reason is that both V2VNet and when2com employ a scalar to denote the agent-to-agent attention, which cannot distinguish which region is more informative; while DiscoNet can adaptively find beneficial region in a cell-level resolution; see the visualization of matrix-valued edge weights in Fig. 6 (d)-(f).

Comparison of with and without teacher. To verify the effectiveness of knowledge distillation, Fig. 7 compare the detection output and matrix-valued edge weight of the proposed DiscoNet with and without the regularization from the teacher model. We see that, without knowledge distillation regularization, the matrix-valued edge weight does not reflect informative spatial attention, mostly degenerating to a scalar-valued edge weight. Thus, the detector loses five safety-critical objects.

4.4 Ablation study

Effect of collaboration strategy. Table 3 compares the proposed method with five baselines using different intermediate collaboration strategies. *Sum* simply sums all the intermediate feature maps. *Average* calculates the mean of all the feature maps. *Max* selects the maximum value of all the feature maps at each cell to produce the final feature. *Cat* concatenates the mean of others' features with its own feature, and then uses a convolution layer to halve the channel number. *Weighted Average* appends one convolutional layer to our edge encoder Π, to generate a scalar value as the agent-wise edge weight. We use a matrix-valued edge weight to model the collaboration among agents. Without knowledge distillation, *Max* has achieved the best performance probably because it ignores the noisy information with lower value. We see that with the proposed matrix-valued edge weight and knowledge distillation, DiscoGraph significantly outperforms all the other collaboration choices.

Effect of knowledge distillation. In Table 3 we investigate the versions with and without knowledge distillation, We see that i) for our method, the knowledge distillation can guide the learning of collaboration graph, and the agents can work collaboratively to approach the teacher's performance. Quantitatively, knowledge distillation yields an improvement of 7.1%/6.3% in AP@0.5/0.7; ii) for *max* and *average* without a learnable module during collaboration, knowledge distillation has no impact; and iii) for *sum* and *cat*, their performances are improved a little bit, as knowledge distillation can influence the feature abstraction process. Table 2 further shows the detection performances when we apply knowledge-distillation regularization at various network layers. We see that i) once we apply knowledge distillation to regularize the feature map, the proposed method achieves significant improvement; ii) by only regularizing the output of encoder ($\mathbf{H}_i^s$), the precision can be improved to 59.0/53.4 at IoU=0.5/0.7; and iii) applying more regularization has slight influence.

5 Conclusion

We propose a novel intermediate-collaboration method, called distilled collaboration network (DiscoNet), for multi-agent perception. Its core component is a distilled collaboration graph (DiscoGraph), which is novel in both the training paradigm and the edge weight setting. DiscoGraph is also pose-aware and adaptive to perception measurements, allowing multiple agents with the shared DiscoNet to collaboratively approach the performance of the teacher model. To validate, we build a large-scale multi-agent 3D object detection dataset based on CARLA and SUMO, and the experiments show that DiscoNet achieves appealing performance-bandwidth trade-off.

References

[1] Rohan Anil, Gabriel Pereyra, Alexandre Passos, Robert Ormandi, George E Dahl, and Geoffrey E Hinton. Large scale distributed neural network training through online distillation. *arXiv preprint arXiv:1804.03235*, 2018. 2

[2] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020. 6

[3] Qi Chen, Sihai Tang, Qing Yang, and Song Fu. Cooper: Cooperative perception for connected autonomous vehicles based on 3d point clouds. In *IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 514–524, 2019. 1

[4] Siheng Chen, Baoan Liu, Chen Feng, Carlos Vallespi-Gonzalez, and Carl K. Wellington. 3d point cloud processing and learning for autonomous driving: Impacting map creation, localization, and perception. *IEEE Signal Processing Magazine*, 38:68–86, 2021. 1

[5] Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. Tarmac: Targeted multi-agent communication. In *International Conference on Machine Learning*, pages 1538–1546. PMLR, 2019. 5

[6] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017. 2, 6

[7] Jakob N. Foerster, Yannis M. Assael, N. D. Freitas, and S. Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *NIPS*, 2016. 5

[8] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012. 1

[9] Ramyad Hadidi, Jiashen Cao, Matthew Woodward, Michael S Ryoo, and Hyesoon Kim. Distributed perception by collaborative robots. *IEEE Robotics and Automation Letters*, 3(4):3709–3716, 2018. 5

[10] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 2

[11] Yedid Hoshen. Vain: Attentional multi-agent predictive modeling. *arXiv preprint arXiv:1706.06122*, 2017. 5

[12] Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. *arXiv preprint arXiv:1805.07733*, 2018. 5

[13] Saifullah Khan, Franz Andert, Nicolai Wojke, Julian Schindler, Alejandro Correa, and Anton Wijbenga. Towards collaborative perception for automated vehicles in heterogeneous traffic. In *International Forum on Advanced Microsystems for Automotive Applications*, pages 31–42. Springer, 2018. 5

[14] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. Recent development and applications of sumo-simulation of urban mobility. *International journal on advances in systems and measurements*, 5(3&4), 2012. 6

[15] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4558–4567, 2018. 1

[16] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *International journal of computer vision*, 128(2):261–318, 2020. 1

[17] Yen-Cheng Liu, Junjiao Tian, Nathaniel Glaser, and Zsolt Kira. When2com: multi-agent perception via communication graph grouping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4106–4115, 2020. 1, 4, 5, 6, 7

[18] Yen-Cheng Liu, Junjiao Tian, Chih-Yao Ma, Nathan Glaser, Chia-Wen Kuo, and Zsolt Kira. Who2com: Collaborative perception via learnable handshake communication. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6876–6883. IEEE, 2020. 1, 4, 5, 6, 7

[19] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 3569–3577, 2018. 1

[20] Seyed Mojtaba Marvasti-Zadeh, Li Cheng, Hossein Ghanei-Yakhdan, and Shohreh Kasaei. Deep learning for visual tracking: A comprehensive survey. *IEEE Transactions on Intelligent Transportation Systems*, 2021. 1

[21] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International conference on artificial neural networks*, pages 52–59. Springer, 2011. 3

[22] Shervin Minaee, Yuri Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *IEEE transactions on pattern analysis and machine intelligence*, PP, 2021. 1

[23] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014. 2

[24] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–779, 2019. 1

[25] Shaoshuai Shi, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. *IEEE transactions on pattern analysis and machine intelligence*, 2020. 2

[26] Weijing Shi and Raj Rajkumar. Point-gnn: Graph neural network for 3d object detection in a point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1711–1719, 2020. 2

[27] Yoav Shoham and Kevin Leyton-Brown. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, 2008. 5

[28] Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Learning when to communicate at scale in multiagent cooperative and competitive tasks. *ArXiv*, abs/1812.09755, 2019. 5

[29] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. *arXiv preprint arXiv:1605.07736*, 2016. 5

[30] Nicholas Vadivelu, Mengye Ren, James Tu, Jingkang Wang, and Raquel Urtasun. Learning to communicate and correct pose errors. *arXiv preprint arXiv:2011.05289*, 2020. 6

[31] Tsun-Hsuan Wang, Sivabalan Manivasagam, Ming Liang, Bin Yang, Wenyuan Zeng, and Raquel Urtasun. V2vnet: Vehicle-to-vehicle communication for joint perception and prediction. In *Proceedings of the European Conference on Computer Vision*, pages 605–621. Springer, 2020. 1, 2, 4, 5, 6, 7

[32] Yufeng Yue, Chunyang Zhao, Mingxing Wen, Zhenyu Wu, and Danwei Wang. Collaborative semantic perception and relative localization based on map matching. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020. 5

[33] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018. 2

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See the discussion in line 217-218 in Section 3.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See the supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4.1 and the supplemental material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We have not repeated experiments many times to get error bars since experiments of 3d object detection on large scale datasets is time-consuming.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 4.2.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 4.1.
   (b) Did you mention the license of the assets? [Yes] See Section 4.1.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See Section 4.1 and the supplemental material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We generate the data using the open-sourced tool and the owners consent to all the public use for research purposes.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] Our data is synthesized using the open-sourced tool, so there is no real-world personally identifiable information, nor offensive content.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]