MultiVTrain: Collaborative Multi-View Active Learning for Segmentation in Connected Vehicles

Huiye Liu
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia, USA
huiyeliu@gatech.edu

Douglas M. Blough
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia, USA
doug.blough@ece.gatech.edu

***Abstract*—While deep learning has brought promising advances to semantic segmentation tasks for autonomous vehicles, the performance strongly depends on the coverage of collected data and the quality of annotations. To overcome the barrier of insufficient high quality training data covering a complex range of vehicular scenarios, in this paper, we propose a multi-view-based active learning framework (MultiVTrain), which enables the vehicles to collaboratively generate training data and accurate labels without querying remote human annotators. As images captured by RGB cameras are vulnerable to occlusion and limited field-of-view, a novel multi-view prediction transfer scheme is introduced to leverage sensor data fusion and transfer predictions of one view to another. This allows information from different views to be aggregated, which improves the quality of the generated annotations. Extensive evaluation results demonstrate that our proposed MultiVTrain framework outperforms other active learning baselines by $\sim$ 9%, and passive supervised learning baselines trained with ground truth labels by $\sim$ 2.5%, for the same training set size.**

***Index Terms*—Autonomous vehicles, connected vehicles, active learning, online learning, multi-view collaborative perception.**

I. Introduction

Semantic segmentation is of great significance for autonomous vehicles to understand the environment [1], where each pixel in an image is marked with categorical labels, representing drivable area, pedestrians, traffic participants, buildings, etc. Recent advances in deep learning have led to the development of semantic segmentation using convolutional neural networks [2]–[4]. However, these methods are focused on centralized supervised learning and, hence, their performance hinges on acquiring a huge amount of well-labeled data for training. Creating large labeled datasets is prohibitively expensive as it requires human annotators to accurately trace segment boundaries and produce pixel-level labels. Moreover, it requires not only collecting traffic scene images with sufficient variations in terms of lighting conditions, weather, terrain, environment, etc., but also incurs very high overheads due to the tremendous amount of data that needs to be stored and transferred.

Active learning (AL) has proven to be a powerful technique to improve data efficiency for supervised learning methods, where the key idea is that a machine learning algorithm can achieve better performance with fewer training labels if it is allowed to choose the data from which it learns [5]. Prior works have demonstrated the great potential active learning can add to the training performance as well as efficiency [6]–[9]. However, most of this work assumes an oracle labels the query data samples, which is impractical in vehicular networks. In addition, as vehicles capture data in a streaming style, pool-based re-training is very expensive and can hardly be accomplished by vehicles locally. Offloading all data and training tasks to a centralized server introduces other challenges such as scalability and bandwidth [10]. Therefore, a scalable and decentralized active learning framework without an oracle is needed, so that the vehicles can select data and train locally in an online fashion.

Advances in connected and autonomous vehicles allow vehicles to exchange information through various communication protocols (V2V, V2I, V2X) and be equipped with powerful sensors (lidar, camera, etc.) and processing units. This makes it possible to leverage sensor data fusion and data aggregation from nearby vehicles with multiple viewing angles, and potentially address the challenges of occlusion, low resolution due to long distance, and non-line-of-sight effects. In this paper, we propose a multi-view based collaborative active learning framework (MultiVTrain) addressing the above challenges, where a group of vehicles can perform online learning by cooperating to choose informative instances and automatically annotate them without human help, thereby enabling the creation of accurate models locally without support of a centralized cloud. In specific, our contributions include:

- A multi-view prediction transfer scheme to align different views from multiple vehicles via leveraging sensor data fusion and facilitate cooperative generation of pseudo labels.
- A collaborative online annotation algorithm, which replaces human annotation by synthesizing the predictions generated by neighboring vehicles' local models and correlated multi-views. As a result, data annotation and model updates can be completed locally without support of a centralized server.
- A data selection scheme that accounts for data informativeness, cross-view diversity, and the accuracy of the current model to save local computational resources by selecting the specific instances that have the best chance to improve model performance.
- An implementation and of our MultiVTrain framework

for multi-class semantic segmentation is built on top of Carla. Extensive evaluation are conducted to demonstrate the effectiveness of the approach.

II. Related Work

Deep learning based semantic segmentation is one of the important techniques to realize driving scene perception and localization [1]. Networks like SegNet [11], IC-Net [12]are mainly encoder-decoder architectures with a pixel-wise classification layer. These are based on building blocks from some common network topologies, such as PSPNet [13], VGG-16 [14], and ResNet [15]. Though these approaches show promising performance, they are pool-based passive learning methods that have the limitations of requiring highly accurate data labels and enormous re-training costs.

Active learning (AL), which aims at finding the minimum amount of labeled data to achieve a certain performance, has been considered as a promising approach to address the data and efficiency challenges. [16] provides a thorough review of classical AL literature. How to select the next batch of query samples to be labeled is a focus in recent image-based AL works. Three major approaches have been proposed: uncertainty-based, diversity-based, and expected model change [17]. Since classical AL methods require an oracle to label sampled data, which hinders practical deployment, replacing oracle-based labeling with other noisy label generation methods has also been studied in recent AL works [18]–[20].

Most existing noisy label generation methods have heavy memory and compute requirements, which is a problem for deployment in decentralized vehicular networks. Considering the unique environment that vehicles operate in, a light-weight and resource efficient AL framework is necessary. To address this challenge for the classification task, Abdellatif, et al. proposed a cooperative pseudo label generation scheme and a data selection scheme based on data quality as well as diversity [21]. However, their model accuracy metric applied to label generation may be biased by the selected test set. Performing well on one data set does not guarantee good performance on new data sets, especially for vehicular applications, where different road scenarios are virtually unlimited. Li, et al. take advantage of the multi-view effect to address the partial occlusion issue in vehicle detection [9]. However, their set up is specific to the detection problem and is not easily adapted for the segmentation task, and their approach also assumed human annotators for label generation.

We propose MultiVTrain, a novel learning framework based on AL, for improving semantic segmentation with unseen scenarios in vehicular networks. MultiVTrain uses a multi-view prediction transfer method to improve pseudo label quality for a single vehicle. It also employs a collaborative method based on depth information to intelligently integrate pseudo labels from nearby vehicles to obtain high-quality annotations for sampled data. In this way, the segmentation model can be updated locally with newly collected data without human annotation. MultiVTrain also improves efficiency and saves memory by selecting only the most significant data instances

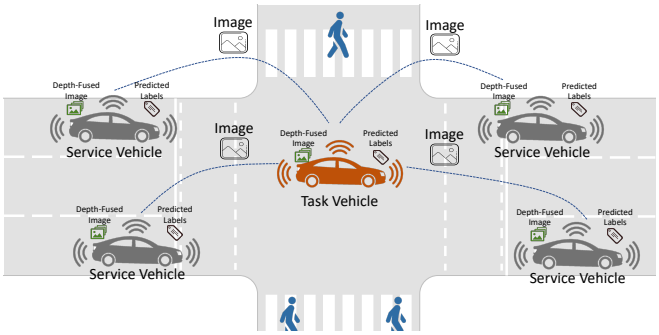


Fig. 1: MultiVTrain System Model Overview

to be added to the training set using a combination of data uncertainty, cross-view diversity, and local model prediction.

III. System Model and Preliminaries

In this section, we first present our system model in III-A. As depth-fused images play an important role in our proposed approach, the pre-processing and fusion procedures we adopt are introduced in III-B.

A. System Model

As shown in Fig. 1, our proposed system model considers a group of vehicles, where each is equipped with an RGB camera, a depth sensor (e.g. lidar, depth camera), a pre-trained machine learning model, and local processing units capable of performing sensor data fusion and local machine learning model updates. We refer to the vehicles that find interesting images to learn and initiate a round of collaborative active learning as *TaskVehicles* (TVs). A TV is shown in orange in Fig. 1. We refer to the vehicles that are within communication range of a task vehicle and are capable of serving a collaborative learning task as *ServiceVehicles* (SVs). SVs are shown in grey in Fig. 1. We assume basic communication among vehicles is supported to allow system-level beacon messages as well as application messages. The periodic beacon messages allow vehicles to discover neighboring vehicles. The application messages allow vehicles to exchange sensor data, calibrated sensor parameter, vehicle location, velocity and moving direction.

B. Depth-Fused Images

Sensor fusion is used to aggregate correlated multi-view images from nearby vehicles for pseudo labeling. As shown in Fig. 2, we obtain depth-fused images by fusing information from depth sensors with 2D RGB images in an early fusion style. Since our proposed approach does not have limitation on the choices of depth sensors, we use point clouds, which can be easily obtained, to illustrate the process of depth-fused image generation. Following the standardized formulation in [22], let each point in a 3D point cloud be represented by $[x, y, z, 1]^T$, where x, y, and z are the coordinates of the point. Note that, given the coordinates of the depth sensor, x_s, y_s, and z_s, the depth information d for the point $[x, y, z, 1]^T$ can be easily calculated. For simplicity, we assume that the d value is stored

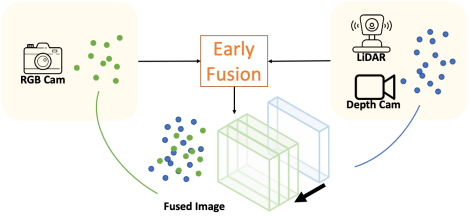


Fig. 2: Depth-Fused Image

as part of the data record of each point. Let a pixel in an image plane be represented by $[u, v, 1]^T$, where u and v denote the row and column position of the pixel, respectively.

As the relative position between depth sensor and camera can be acquired through calibration, the point cloud can be projected to the image pixel array according to:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{1}$$

where K, R, and t denote the intrinsic matrix, rotation matrix, and translation vector of the RGB camera, respectively [22]. By applying Eq. 1 to every point in the point cloud, point cloud points, along with their depth values, are mapped to corresponding pixels in the 2D image.[1] In this way, a depth map D of the same dimension as an image $\mathcal{I}$ captured by the RGB camera is obtained. Through matrix concatenation, D and I are combined to become the depth-fused image $\mathcal{D} = \mathcal{I}|D$, where the value in each cell of $\mathcal{D}$ contains red, green, blue, and depth values. During the concatenation, pixels without projected depth info will be automatically assigned ∞ as a depth value. To simplify the discussion, everywhere we use the term "image" in the following sections, we refer to a 2D RGB image.

IV. Multi-View Prediction Transfer

As obtaining human annotation is impractical as well as expensive in vehicular networks, the multi-view prediction transfer (MPT) scheme is proposed to improve the quality of generated pseudo labels. Inspired by the image-based shape-from-silhouette [23] and 3D shape belief transfer proposed in [24], we use depth-fused images to fuse multi-view information for better pseudo label generation. For ease of presentation, we simplify the terminology so that wherever we refer to the "label of an image", we are referring to the label matrix that has a label for each pixel of the image.

1) How does MPT work: Consider a scenario that v_i and v_j are moving on the same road but in different lanes. As depicted in Fig. 3(a) and Fig. 3(c), we can see that v_i is in front of v_j and the black vehicle shown in v_j's view is indeed v_i. Though the locations of v_i and v_j are different, they share some common objects in their views. The MPT task goal here is to let v_j produce pseudo labels for v_i's image, denoted by $\mathcal{I}_i^t$, captured at time step t. Assume v_j is equipped with a local model that can generate per-pixel prediction $\phi_{j,n}$ for every pixel n on its input image. Thus, two predictions can be made by v_j: 1) $\phi_{j,n}(\mathcal{I}_i^t)$, prediction of v_i's image at time step t, and 2) $\phi_{j,n}(\mathcal{I}_j^t)$, prediction of v_j's corresponding image captured at time step t. As the second prediction is made on a different view (image), we need to transfer the second one to the first one's image plane, so that we can aggregate the predictions from these two different views. Following the procedure introduced in Section III-B, both v_i and v_j are able to obtain the depth-fused image $\mathcal{D}_i^t$, $\mathcal{D}_j^t$ based on their local sensor data. As stated in Section III-A, we assume that $\mathcal{D}_i^t$, K_i, and location of v_i is known to v_j upon receiving v_i's application message. Then, v_j can easily reconstructs the rotation matrix R_i and translation matrix t_i of v_i. Therefore, given the depth information incorporated in the depth-fused image, v_j is able to transfer the per-pixel prediction $\phi_{j,n}(\mathcal{I}_j^t)$ to the view of v_i (the plane of $\mathcal{I}_i^t$). by:

$$[\phi_{j,n}(\mathcal{I}_i^t \leftarrow \mathcal{I}_j^t)|\epsilon] = [\phi_{j,n}(\mathcal{I}_j^t)|D_j^t]K_j^{-1} \begin{bmatrix} R_j & t_j \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} R_i & t_i \\ 0 & 1 \end{bmatrix} K_i \tag{2}$$

where ϵ represents the redundant numbers generated by matrix transformation. Hence by dropping the ϵ term, the transferred prediction $\phi_{j,n}(\mathcal{I}_i^t \leftarrow \mathcal{I}_j^t)$ is obtained as shown in Fig. 3(e). v_j now obtains two sets of predictions toward the same input image $\mathcal{I}_i^t$, incorporating information from two different views. By extending this procedure to a group of neighboring vehicles with diverse views, a fuller understanding of the common objects can be obtained.

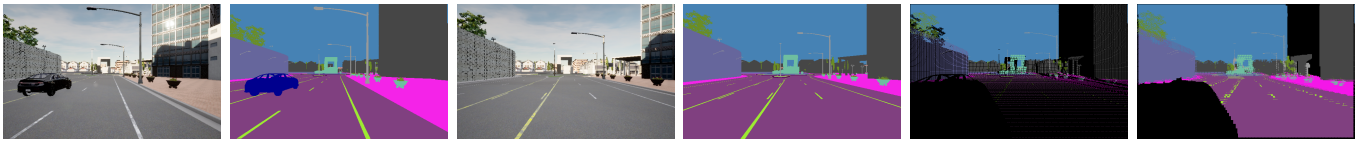
2) MPT results improvement: As we can see from the figure that the depth information is missing at certain pixel locations (black in Fig. 3(e)), this is due to either out of range objects or the sparsity of captured point clouds. If the point clouds are too sparse, the benefit of this scheme will be deteriorated, because there would be little information to correctly correlate pixel locations in diverse views. We follow the interpolation method introduced in [25] to upsample the depth-fused images, so that the depth-fused images with denser depth information are obtained. For the pixel location whose depth value is still missing after the cure, we treat the pixel as out of scope pixel and fill the depth value with $+\infty$.

3) Adaptation to other image-based machine learning tasks: Moreover, this multi-view shape transfer scheme can be easily adapted to bounding box object detection, by replacing the per-pixel segmentation prediction to the bounding box prediction or sample points with depth information inside the bounding box.

V. Online Active Learning Framework

In this section, we provide the details of the online active learning framework in our proposed MultiVTrain methodology for the application of semantic segmentation. While we believe the approach can be extended fairly easily to other tasks such as object detection, we leave extensions as future work and focus on semantic segmentation from here on. From the high-level perspective, we address the challenge of expensive data

[1] All resulting points $[u, v]$ that fall outside the boundary of the image plane are discarded.



(a) Image $\mathcal{I}_j^t$ Captured by v_j at time step t (b) Carla-Generated Ground Truth of $\mathcal{I}_j^t$ (c) Image $\mathcal{I}_i^t$ Captured by v_i at time step t (d) Carla-Generated Ground Truth of $\mathcal{I}_i^t$ (e) Raw MPT results $\phi_{j,n}(\mathcal{I}_i^t \leftarrow \mathcal{I}_j^t)$ (f) Upsampled MPT results $\phi_{j,n}(\mathcal{I}_i^t \leftarrow \mathcal{I}_j^t)$

Fig. 3: Example Scenario of Multi-View Prediction Transfer (MPT)

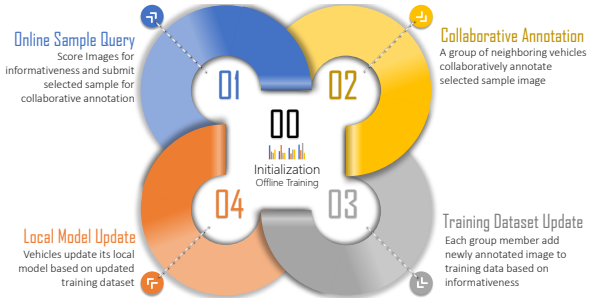


Fig. 4: Online Active Learning Framework Loop

labeling for machine learning in vehicular networks by online active learning, where no human labelers and centralized servers are required. The five-stage loop of our online active learning framework is illustrated in Fig. 4.

We assume an initial segmentation model is pre-trained and installed in each vehicle before leaving the factory. This model is then updated by each round of active learning. During the online sample query stage, a vehicle will estimate the informativeness of the newly captured images. Whenever there is an informative image found, the vehicle will interact with its neighboring vehicles to collaboratively annotate the image based on group members' own images and local model predictions. After group decision of the image labeling, each vehicle in the group will decide based on their own situation whether this labeled image should be added to its training dataset. Each vehicle will update its local model after the updates of its training dataset. However, to reduce computational overhead, we let each vehicle aggregate multiple new annotated images in their training dataset before performing a model update. Details of each stage are provided next.

A. Initialization

In the initialization stage, passive learning consisting of the conventional supervised learning of a multi-class segmentation based on human annotation ("ground-truth" labeling) is performed. The pre-trained base model is produced from M well-labeled data samples. The base model takes an input image $\mathcal{I}$ and outputs a per-pixel object confidence, i.e., $\phi(\mathcal{I}, \omega) \in [0,1]^{N \times C}$, where N is dimension of the output distribution, and C is the number of object classes. This is equivalent to a multi-class segmentation task [26]. We use the standardized sum of pixel-wise cross entropy to measure the segmentation loss:

$$\mathcal{L}_{ce} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C} y_{n,c}\log(\phi_{n,c}) \tag{3}$$

where $y \in \{0,1\}$ is the ground truth label, N denotes the number of pixels in an image, and $y_{n,c}$ and $\phi_{n,c}$ represent the ground truth label and probability prediction for the n_{th} pixel of class c, respectively.

We assume that different brands of vehicles hold different initial models as, in reality, they usually do not share labeled datasets. Moreover, as different vehicles hold different trip histories, even if the initial model is the same at the beginning, the model will be different after certain times of local model updates. Therefore, we assume that the initial models for all vehicles are different in some way.

B. Sample Query

The key reason why active learning can efficiently improve a model with less training data is that it allows the machine learning algorithm to choose the data from which it learns, e.g. by selecting images that the current model does not predict well. Such data is said to be *informative* [5]. Scoring the informativeness of an unlabeled image is, therefore, an important component of selecting new input data to learn from. In our proposed MultiVTrain method, we use the *uncertainty* concept to evaluate the informativeness of an image [16]. Thus, the informativeness scoring $\mathcal{S}$ of an image $\mathcal{I}$ with N pixels is calculated by cross-entropy as:

$$\mathcal{S}(\mathcal{I}) = \frac{1}{N}\sum_{n=1}^{N}\mathcal{H}(\phi_n) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C}\phi_{n,c}\log(\phi_{n,c}) \tag{4}$$

$\mathcal{H}(\phi_n)$ represents the entropy of prediction for one pixel in N, calculated by $\sum_{c=1}^{C}\phi_{n,c}\log(\phi_{n,c})$, where $\phi_{n,c}$ denotes the confidence that pixel n should be predicted as class c.

Though the common approaches use the computed informativeness score to decide which images should be sampled for annotation, yet in vehicular networks, images arrive in a streaming way, where consecutive images in a certain range could score similarly due to the similar view. For instance, if a vehicle stops at a crossing due to red light, the images captured during the wait will be very similar, which need not be learned multiple times, as it wastes the annotation as well as training resources. Therefore we introduce another metric - cross-image diversity, where pixel-wise prediction histogram

plus vehicle location difference are counted, to avoid choosing consecutive images with a similar view. An image is selected as a query sample if and only if both informativeness score and cross-image diversity are larger than certain threshold.

Note that different from other active learning frameworks, we will not train on every queried samples. Considering the relatively limited local resources, we allow the vehicle to decide whether to learn from a sample in the Training Dataset Update stage. More details are provided in Section V-D

C. Online Collaborative Annotation

Though offloading all sampled data to remote cloud and relying on human labeling is considered accurate and reliable, the delay and labor cost is non-negligible. Besides, neighboring vehicles usually hold overlapping objects in their views, where the views usually capture the same object with different angles, distances, and occlusion conditions. Hence, by combining these multiple views together, a fuller observation of these objects can be obtained and better segmentation results can be achieved than using images from one single view. Therefore, instead of resorting to human labeling, MultiVTrain achieves data annotation in a distributed fashion without human intervention by leveraging neighboring vehicles' different local models along with their multi-view depth-fused images.

As described in Sec. III-A, we assume that each vehicle is equipped with a depth sensor and an RGB camera facing the front and that a 3D point cloud obtained from the depth sensor is fused with a 2D color image captured by the RGB camera to produce a depth-fused image. Once a sample image $\mathcal{I}_i^t$ is successfully selected during the sample query stage, its corresponding depth-fused image will be generated. Following the notation used in Section IV, let $\mathcal{D}_i^t = (\mathcal{I}_i^t, d_i^t)$ denote a depth-fused image generated by vehicle v_i in time step t, and $\phi_i(\mathcal{I}_i^t)$ denote the segmentation prediction generated by v_i's local model towards $\mathcal{I}_i^t$. As illustrated in Figure 1, the TV ($v_i \in G$) will form a group G with its neighboring vehicles ($v_j \in G$), and broadcast its depth-fused image $\mathcal{D}_i^t$ and prediction $\phi_i(\mathcal{I}_i^t)$ to the group members (Lines 2-3 of Algorithm 1). Upon receiving TV's data, a SV will need to produce a pseudo label towards $\mathcal{I}_i^t$. Since different vehicles hold different pre-trained local models and each vehicle's travel history is unique, their capability of prediction for different types of objects, lighting condition, traffic scenario, etc. will be different. It is important to aggregate the strength of prediction of diverse models. Therefore, a SV v_j will first compute a prediction directly based on its local model to obtain $\phi_j(\mathcal{I}_i^t)$ (Line 5). However, as a vehicle's local model is not perfectly accurate and could potentially be affected by the view and distance to the components, we use MPT scheme (see Section IV) to obtain a transferred prediction $\phi_j(\mathcal{I}_i^t \leftarrow \mathcal{I}_j^t)$ based on SV's own view. A pseudo label would be generated by synthesizing the local model prediction and MPT generated prediction. (Lines 7-11).

Intuitively, if an object is closer to the vehicle (depth d) or is more centered to the camera (angle α), the object is less likely to be occluded and is more likely to show better resolution in the captured image. Hence, a novel depth-boosted prediction integration scheme is proposed as:

$$\mathcal{L}_{j,n}(\mathcal{I}_i^t) = \gamma_{j,n}\phi_{j,n}(\mathcal{I}_i^t \leftarrow \mathcal{I}_j^t) + (1-\gamma_{j,n})\phi_{j,n}(\mathcal{I}_i^t)),$$
$$\gamma_{j,n} = \frac{\frac{1}{2} - \frac{1}{2}\sin(\frac{|\alpha_{j,n}^t| - |\alpha_{i,n}^t|}{\bar{\alpha}})}{1 + e^{\frac{d_{j,n}^t - d_{i,n}^t}{\bar{d}}}} \tag{5}$$

where $\mathcal{L}_{j,n}$ denotes the pseudo label generated by v_j for pixel n. $\gamma \in [0,1]$ is a weighting coefficient defined to combine the two predictions, which is based on depth (d) as well as angle (α[2]) to the center of the camera. Angle $\alpha_{j,n}^t$, $\alpha_{i,n}^t$, emulate the pixel n's centerness to the camera of v_j and v_i respectively, such that the more center position the pixel locates, the larger the angle becomes. Similarly, $d_{j,n}^t$, $d_{i,n}^t$ represent the depth value at pixel n in $\mathcal{D}_j^t$ and $\mathcal{D}_i^t$ respectively. The smaller the depth is, the content captured by pixel n is closer to the camera (vehicle). $\bar{\alpha}$ and $\bar{d}$ are two hyper-parameters that control the sensitivity to the angle and depth value, where the larger the hyper-parameter is, $\mathcal{L}_{j,n}$ is less sensitive to the corresponding value. In summary, the larger the γ becomes at pixel n, the higher weight will be given to the transferred prediction $\phi_{j,n}(\mathcal{I}_i^t \leftarrow \mathcal{I}_j^t)$, because we believe that at pixel n, v_j's view provides better information than v_i's view, and vice versa. Note there is a special case that when $\gamma_{j,n}$ is as close as to 0.25, the view is as close as to the view of the TV. When $\gamma = 0.25$, the two views are considered the same. When the transferred prediction $\phi_{j,n}(\mathcal{I}_i^t \leftarrow \mathcal{I}_j^t)$ is missing at pixel n, $d_j^n = \infty$ and γ becomes zero, in which only the local model prediction $\phi_{j,n}(\mathcal{I}_i^t)$ will be factored in.

Algorithm 1: Depth-Based Weighted Voting Scheme for Pseudo Label Integration

1 task vehicle: $v_i \in G$, service vehicle: $v_j \in G$;
2 **while** v_i *selects a valid sample image* $\mathcal{I}_i^t$ **do**
3 v_i broadcast the corresponding depth-fused image $\mathcal{D}_i^t$ and prediction $\phi_i(\mathcal{I}_i^t)$;
4 **for** *all available* v_j *receives* $\mathcal{D}_i^t$, $\phi_i(\mathcal{I}_i^t)$ **do**
5 computes prediction $\phi_j(\mathcal{I}_i^t)$ using v_j's local model;
6 apply MPT scheme (Eq. 2) and obtain $\phi_j(\mathcal{I}_i^t \leftarrow \mathcal{I}_j^t)$;
7 **for** *each pixel* n *in* $\mathcal{I}_i^t$ **do**
8 **if** $(n : \mathcal{I}_i^t \leftarrow \mathcal{I}_j^t)$ *exist in* $\mathcal{I}_i^t$ **then**
9 obtain $\mathcal{L}_{j,n}(\mathcal{I}_i^t)$ based on Eq. 5
10 **else**
11 $\mathcal{L}_{j,n}(\mathcal{I}_i^t) = \phi_{j,n}(\mathcal{I}_i^t)$
12 broadcast $\mathcal{L}_{j,n}(\mathcal{I}_i^t)$ to other group members;
13 **for** *all members in* G **do**
14 take v_i's $\phi_i(\mathcal{I}_i^t)$ as $\mathcal{L}_{i,n}(\mathcal{I}_i^t)$;
15 **for** *each pixel* n *in* $\mathcal{I}_i^t$ **do**
16 $\mathcal{L}_{G,n}(\mathcal{I}_i^t) = \frac{1}{|G|}\sum_{v_k \in G}^{|G|} \mathcal{L}_{k,n}(\mathcal{I}_i^t)$
17 go back to the start of WHILE loop;

[2]The angle can be reconstructed from depth-fused image through $\arccos \frac{x_n - x_o}{\sqrt{(x_n-x_o)^2+(y_n-y_o)^2}}$, where x_n, x_o denotes the x axis coordinate of pixel n and camera center o, same as y_n, y_o.

Once a SV obtains per-pixel level pseudo label $\mathcal{L}_{j,n}(\mathcal{I}_i^t)$, it sends its vote to other group members (Line 12). Each group member integrates the pseudo label vote produced by other group members through average voting and obtains the group decision of final label for the sample image $\mathcal{I}_j^t$. As TV is the initiator and does not have another view, its prediction $\phi_i(\mathcal{I}_i^t)$ will be taken as its vote of pseudo label directly. (Lines 13-16).

D. Training Dataset and Local Model Update

Although the image sample is selected by task vehicles instead of service vehicles, the sample might also be informative to learn for the service vehicles or uninformative to learn even for the task vehicle, as the informativeness is decided by estimation. Therefore, we let each vehicle participating in the collaborative annotation stage compare the group annotation results with its own local prediction results. If the prediction difference between local prediction and the group annotation is larger than certain threshold T_{th}, this sample is considered as valuable to learn. In that case, this would also add the data sample into its training dataset.

To avoid catastrophic forgetting, i.e., forgetting old tasks in the presence of more recent tasks and to save computation resources, we update the local model once 64 new informative data samples are captured. We follow the online continual learning paradigm as discussed in [27]. To be more specific, we adopt the naive rehearsal method [28], where a small replay buffer is built to store a fraction of previous data randomly. While conducting a new round of model update, each mini-batch is constructed by an equal amount $(8/8)$ of new data and the rehearsal data[3].

VI. Simulation Setup and Data Collection

A. Experiment Setup

The evaluation is performed using the Carla simulator [29], which supports a variety of towns, driving scenarios, types of sensors, and ground-truth label generation. Unlike other data sources, Carla allows us to generate images of the same objects from multiple vehicles at different locations at the same time, which is necessary for evaluating our proposed approach. Details of the experimental setup are provided below.

1) Simulation Scenario: A group of five vehicles was simulated in Carla environment. In order to let them stay within communication range so that groups can be formed, the built-in PDE control is leveraged to limit the maximum pair-wise distance among the vehicles to be 250m. Since we do want to simulate different aspects of multi-view effects, nothing else was controlled other than the pair-wise distance. Thus, the relative position, direction, speed, etc. between the vehicles were dynamic. An initial model, a depth camera[4], and an RGB camera were attached to each vehicle and calibrated in the same way, so that with the application messages as described in Sec. III-A, a vehicle was able to calculate neighboring vehicles' rotation and translation matrices easily. At run time, each vehicle captures an image every 100 ms and tries to find a sample image to learn. Once a sample is found, the group of vehicles will cooperatively annotate the sample and update their training sets and local models as needed.

2) Dataset Collection: Four training sets, one validation set, and one test set were collected using Carla. Training set 1 (TS 1: 1400 images) and the validation set (200 images) for building the passive learning models was generated by a single vehicle traveling in the built-in town maps (Town01 and Town07), where groundtruth labels "-pagt") are auto-generated by Carla. The test set, which consists of 400 images, was collected through the same procedure but from Town05 instead, so that we can make sure that no vehicles have seen similar scenes before to avoid unfair comparisons. Training set 2 (TS 2: 448 images) is collected through simulation of active learning (AL), where two sets of labels are obtained: auto-generated groundtruth labels "-acgt") and pseudo labels ("-acpl") generated by a module implementing our proposed MultiVTrain procedure. Training sets 3 and 4 (TS 3: 448 images and TS 4: 448 images) of the same size as TS 2 are generated under the exact same simulation set up but are used to run different AL frameworks for comparison. Implementation details are provided in Sec. VI-B. Finally, all images are collected in the form of RGB images with the resolution of 680x420 pixels, while other information such as depth-fused images and vehicle transformations are also recorded for TS 2, in order to be able to repeat the experiments.

TABLE I: Model Training Settings Overview (gt represents "groundtruth", pl represents "pseudo label")

Models	Training Method	Training Set
PSP-pagt	Passive, Offline	1148 (TS 1, GT)
Deepv3-pagt	Passive, Offline	1148 (TS 1, GT)
PSP-acgt	Active, Offline	700 (TS 1, GT) + 448 (TS 2, GT)
Deepv3-acgt	Active, Offline	700 (TS 1, GT) + 448 (TS 2, GT)
PSP-acpl	Active, Offline	700 (TS 1, GT) + 448 (TS 2, PL)
Deepv3-acpl	Active, Offline	700 (TS 1, GT) + 448 (TS 2, PL)
ACMV-acpl	Active, Online	700 (TS 1, GT) + 448 (TS 3, PL)
ACWA-acpl	Active, Online	700 (TS 1, GT) + 448 (TS 4, PL)
MultiV-acpl	Active, Online	700 (TS 1, GT) + 448 (TS 2, PL)
MultiV-acgt	Active, Online	700 (TS 1, GT) + 448 (TS 2, GT)

B. Training Details

Two MultiVTrain models and eight baseline models are trained with different settings to provide a thorough evaluation. All models are built using the PyTorch framework and trained using a single NVIDIA-RTX2080Ti GPU. The Adam optimizer was adopted and all models were trained for 40 epochs and a batch size of 16. The learning rate was set to 1e-4. Details of each training method are provided below.

1) Training for MultiVTrain: PSPNet was selected as our segmentation network and the two hyper-parameters $\bar{\alpha}$ and $\bar{d}$ were set to 1.5 and 500 empirically. Each initial model was

[3]We use the combination of $(8, 8)$ here due to the GPU limit. Though the $(64/64)$ setup listed in the paper is more ideal, the $1:1$ ratio between new data and the rehearsal data in each batch size is more important.

[4]As discussed earlier, either depth camera or lidar can be used to provide depth info and evaluate our proposed approach. The raw data captured by lidar is 3D point cloud while the raw data captured by depth camera is encoded in 2D matrix format, which can be reconstructed to 3D point cloud if needed. Thus, depth camera is used to save I/O delay and computer storage. However, which sensor is used should not impact the performance of the approach.

trained with 700 randomly sampled images from TS 1 and data augmentation including horizontal flip, random rotation, random crop, Gaussian noises were used. The same validation set was used for all initial models. During the AL stages, no data augmentation was done and the batch size of 16 was split to $(8/8)$ as described in Section V-D. Simulation ended after 448 samples were added to a training set, where 228 samples were each collected from Town01 and Town07. To simplify the evaluation process and fairly compare performance with baseline approaches, we recorded all sampled data and its corresponding parameters, so that other approaches could be trained on the exact same dataset. The MultiVTrain model trained with pseudo labels is denoted by "MultiV-acpl", while the model trained with ground truth labels is denoted by "MultiV-acgt".

2) Training for Baselines: We compare our approach with different baseline algorithms by adapting two existing supervised learning methods, PSP [13] and DeepLabv3 [30], and the active learning schemes in [18], [21].

a) *Supervised Learning Baselines*: To evaluate how each stage of our active learning framework performs, we trained the two supervised learning methods in offline style, which generates the best predictor by learning on the entire training data set at once. With the PSP method, we randomly selected 1148 images from TS 1 with ground truth labels to produce the PSP-pagt model. Then, 700 images randomly sampled from TS 1 plus 448 random images from TS 2 with pseudo labels generated with our AL approach were used to produce PSP-acpl. Finally, the same set of 700 TS 1 images and 448 TS 2 images but with ground truth labels were used to obtain PSP-acgt. In the exact same way as just described for PSP, we generated three models using the DeepLabv3 method, which are denoted by Deepv3-pagt, Deepv3-acpl and Deepv3-acgt.

b) *Active Learning Baselines*: To compare MultiVTrain with other AL approaches, two AL baselines were implemented and trained in an online style as in our approach, where selected data is used to update the best predictor for future data at each step, instead of retraining on the entire new dataset. We used the uncertainty based data selection and majority voting (MV) pseudo label generation method, as proposed in [18], and trained in the exact same manner as in MultiV-acpl to produce the ACMV-acpl model. Similarly, we replaced our proposed data selection and collaborative annotation methods with the quality-diversity selection (QDS) and weighted average (WA) integration method, as proposed in [21], to obtain ACWA-acpl.

A summary of the training methods and applied training sets for all models is provided in Table I.

VII. Evaluation

As different image segmentation tasks serve different application goals, we use the standardized Intersection-Over-Union (IoU), also referred to as Jaccard Score, to evaluate the per-class prediction performance. The IoU is calculated as:

$$\mathrm{IoU} = \frac{n_{cc}}{n_{cc} + \sum_{\eta \neq c}^{C} (n_{\eta c} + n_{c\eta})} \tag{6}$$

where n_{cc} presents the number of pixels which are labeled as class c and predicted as class c (True Positives), $n_{\eta c}$ is the number of pixels which are not labeled as class c but predicted as class c (False Positives); similarly, $n_{c\eta}$ is the number of pixels which are labeled as class c but are not predicted as class c (False Negatives). The mean over the per-class IoUs, denoted by mIoU, is used to quantify the overall segmentation performance. A quantitative comparison is shown in Table II. Ten models with different configurations are tested with the same unseen test set (collected from Town05 as described in Sec. VI-A2). Per-class IoUs are shown in the first 13 data columns and the mIoU is provided in the last column.

A. Performance of Different Aspects of MultiVTrain

We begin by evaluating three important aspects of our proposed MultiVTrain framework: 1) whether it is effectively selecting data samples that improve the model; 2) how its generated annotations (pseudo labels) compare to ground truth; and 3) whether it is vulnerable/sensitive to the common forgetting issue that affects some AL approaches.

1) Effectiveness of Sample Data Selection: PSP-pagt and Deepv3-pagt are both popular supervised learning models trained in passive style where training samples are not selected as in AL (see Table I). In order to determine how well our data selection method works, we compare these approaches to PSP-acgt and Deepv3-acgt, which are trained using the data selected by our method. From Table II, we see that the mIoU of PSP-acgt is 6.8% higher than that of PSP-pagt and the mIoU of Deepv3-acgt is 6.2% higher than that of Deepv3-pagt. For classes that are not predicted well by the passive methods, e.g. traffic light, pedestrian, and car, PSP-acgt and Deepv3-acgt outperform PSP-pagt and Deepv3-pagt by a larger margin, up to 14.8%. This demonstrates that our approach to identifying informative data to learn from, rather than simply learning more data, is effective at improving model accuracy.

2) Effectiveness of Collaborative Annotation: MultiV-acgt is trained like MultiV-acpl except for using ground truth labels instead of our collaborative annotation procedure that produces pseudo labels. From Table II, we see that the mIoU of MultiV-acgt is only 1.5% higher than MultiV-acpl, which shows that our proposed collaborative annotation achieves excellent performance with the pseudo labels it produces. Performance across the individual classes is fairly even, with differences in the range of 0.1% to 4.3% between ground truth labels and pseudo labels. However, even for the worst case with our pseudo labeling ("car (all types)"), ground truth labels produce only 4.3% better accuracy than our pseudo labels, which is not a huge margin. Overall, these results show that the pseudo labels produced by our MPT and depth-boosted integration scheme perform well in that they produce close to ground truth performance.

3) Impact of Online Model Updates: With online learning, models are updated in a faster and more resource-efficient manner, which is particularly important for vehicular networks. However, the "forgetting" problem can occur with online learning, which happens when the model learns new

TABLE II: Quantitative Results Comparison with Multiple Baselines

Methods	Classes (IoU)													Average (mIoU)
	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	pedestrian/rider	car (all types)	
PSP-pagt	0.956	0.778	0.804	0.595	0.601	0.603	0.639	0.757	0.891	0.609	0.936	0.642	0.735	0.734
Deepv3-pagt	0.953	0.769	0.799	0.592	0.596	0.595	0.621	0.753	0.887	0.603	0.929	0.636	0.733	0.728
PSP-acgt	0.980	0.845	0.906	0.631	0.632	0.658	0.697	0.775	0.922	0.632	0.956	0.737	0.822	0.784
Deepv3-acgt	0.979	0.832	0.874	0.627	0.625	0.643	0.682	0.779	0.919	0.628	0.954	0.692	0.809	0.773
PSP-acpl	0.987	0.801	0.844	0.613	0.612	0.629	0.663	0.776	0.910	0.624	0.954	0.665	0.758	0.756
Deepv3-acpl	0.977	0.796	0.838	0.604	0.609	0.605	0.657	0.770	0.902	0.622	0.955	0.661	0.752	0.750
ACMV-acpl	0.884	0.686	0.741	0.536	0.545	0.578	0.598	0.692	0.839	0.561	0.874	0.584	0.674	0.676
ACWA-acpl	0.941	0.722	0.763	0.549	0.551	0.588	0.601	0.717	0.840	0.579	0.882	0.596	0.685	0.693
MultiV-acpl	0.977	0.792	0.824	0.603	0.614	0.617	0.657	0.776	0.903	0.621	0.951	0.659	0.742	0.749
MultiV-acgt	0.978	0.802	0.846	0.611	0.620	0.634	0.671	0.787	0.912	0.626	0.952	0.671	0.774	0.760

unseen data and gradually forgets about important information learned in the past. By integrating the rehearsal-C approach [28] of training with both new and old data into our AL framework, the negative impacts caused by online learning can be minimized. By comparing results of PSP-acpl and Deepv3-acpl with MultiV-acpl (or PSP-acgt and Deepv3-acgt with MultiV-acgt), however, we do see that offline model training does perform slightly better than with online training, albeit at a very high cost in terms of speed and efficiency.

B. Comparison of MultiVTrain and AL Baselines

As mentioned earlier, we adapted two active learning baseline approaches to our problem setting and compared them to MultiVTrain. As shown in Table II, MultiV-acpl achieves 8.1% to 10.8% higher mIoU than ACMV-acpl and ACWA-acpl, with large margins (above 5%) in each per-class improvement, except for pole. This is mainly because both ACMV-acpl and ACWA-acpl neglect the importance of distance to the objects when determining the best pseudo labels. Also, though ACWA-acpl factors in the rated accuracy of the initial model, this approach does not adapt well to new never-before-seen data. Performing well on a limited test set does not mean it will achieve comparable performance on other test sets. As described earlier, we used Town01 and Town07 to generate training data, while leaving the unseen dataset collected from Town05 for testing, which we believe is representative of how active learning in vehicles will occur in practice.

C. MultiVTrain Performance with Varying Parameters

In order to study MultiVTrain performance and conduct fair comparisons with multiple baselines, we fixed several parameters in the prior results. Here, we study the approach's performance as some of those parameters are varied.

1) Model Accuracy vs. Group Size: As we replace the human annotator with a group of vehicles with multiviews, the group size affects MultiVTrain's performance. We varied the group size from 2 to 8 with the remaining parameters unchanged. Selected classes are shown in Fig. 5. We see that as the number of group participants grows, the performance increases. However, the model accuracy improves fastest when going from 3 up to 6 vehicles, while improvement from 2 to 3 and beyond 6 is much smaller. Moreover, the smaller

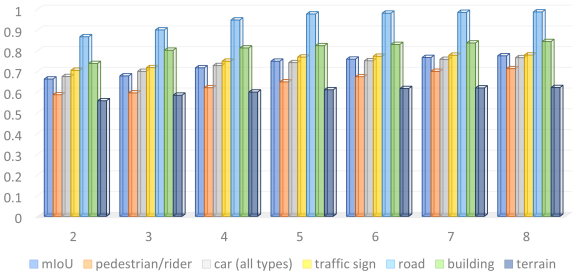


Fig. 5: Model Accuracy (mIoU, IoU) vs. Group Size

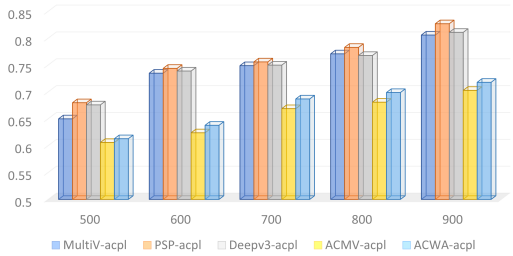


Fig. 6: Model Accuracy (mIoU) vs. Initial Training Set Size

size objects such as pedestrian/rider, traffic sign, and car (all type) show close to linear growth as the group size increases, while larger objects like road and building show more IoU improvement when the group size is increased from 2 to 4 and 2 to 3. These results show that it is important to have around 4 vehicles within a group to provide sufficient view diversity and that gains are smaller with only 2–3 vehicles.

2) Model Accuracy vs. Initial Training Set Size: Intuitively, better accuracy of the initial model could lead to generated annotations with higher quality. As the accuracy of the initial model is mainly decided by the size of the training set, we evaluate how the initial model affects the training results by varying the initial training set size from 500 to 900, leaving the other portion of 448 images with -acpl labels from TS 2 unchanged. Four other models are tested along with MultiVTrain and the results are shown in Fig. 6. We observe from the figure that as the size of training set grows, the performance for all models grows. Note that ACMV-acpl and ACWA-acpl are

more sensitive to the size of initial model, which implies that both majority voting and weighted average integration method mainly account for the initial model performance. This is also why our approach does not show a big performance drop when $size = 600$, contrary to the other AL approaches. Other than only considering the initial model prediction (heavily relies on initial model accuracy), our proposed method accounts for the multi-view benefits by leveraging the depth information, which enables the vehicle to still improve model performance when the initial model is not at a high quality. This aligns with the current real world application requirement in vehicular networks, where existing dataset is limited and cannot well account for all complex scenarios.

3) Other Interesting Parameters: Other than *group size* and *initial model training size*, we believe the group members' view diversity and the AL stage training set size are also interesting to study for future work. Due to limited space, we cannot evaluate these parameters in detail but a short discussion is provided. a) *Diversity of Views*: as demonstrated in prior computer vision works [24] the more views of the same object that can be obtained, the higher chance the object can be better detected by aggregating the information, which agrees with our evaluation results in Section VII-A2 and Section VII-C1. However, given the same group size, we were not able to evaluate how the diversity of views affects the results. b) *AL Training Set Size*: Though AL is known for its efficiency, how would the performance of MultiVTrain grow as the AL training set grows. Will the current performance trend still hold or will the improvement be even larger if the AL training set grows from 448 to 448M, for example? Since our proposed method allows vehicles to collect data and update the model at run time, it is reasonable to let the vehicles continue learning until the model cannot be further improved. The improvement cap in terms of AL training size is, therefore, another interesting topic for future research.

VIII. Conclusion and Future Work

We presented MultiVTrain, an online AL framework, which allows the vehicles to cooperatively generate training data and corresponding labels without querying remote human annotators. In addition, a novel multiview prediction transfer scheme is proposed to enhance label quality via sensor data fusion and multiview alignment. Future work will study the parameters discussed in Section VII-C3 and extend the robustness of the framework to meet other practical challenges in vehicular networks, e.g. tolerating connection loss and achieving higher task completion rate.

References

[1] S. Grigorescu, B. Trasnea *et al.*, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, 2020.

[2] M. Treml, J. Arjona-Medina *et al.*, "Speeding up semantic segmentation for autonomous driving," in *MLITS, NIPS Workshop*, 2016.

[3] G. L. Oliveira, W. Burgard, and T. Brox, "Efficient deep models for monocular road segmentation," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016.

[4] M. Ullah, A. Mohammed, and F. Alaya Cheikh, "Pednet: A spatio-temporal deep convolutional neural network for pedestrian segmentation," *Journal of Imaging*, 2018.

[5] B. Settles, "Active learning literature survey," 2009.

[6] D. Silver, J. A. Bagnell, and A. Stentz, "Active learning from demonstration for robust autonomous navigation," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012.

[7] S. Sivaraman and M. M. Trivedi, "A general active-learning framework for on-road vehicle recognition and tracking," *IEEE Transactions on Intelligent Transportation Systems*, 2010.

[8] R. K. Satzoda and M. M. Trivedi, "Multipart vehicle detection using symmetry-derived analysis and active learning," *IEEE Transactions on Intelligent Transportation Systems*, 2015.

[9] D. L. Li, M. Prasad, C.-L. Liu, and C.-T. Lin, "Multi-view vehicle detection based on fusion part model with active learning," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[10] Y. Sun, X. Guo, S. Zhou, Z. Jiang, X. Liu, and Z. Niu, "Learning-based task offloading for vehicular cloud computing systems," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018.

[11] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, 2017.

[12] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "Icnet for real-time semantic segmentation on high-resolution images," in *Proceedings of the European conference on computer vision (ECCV)*, 2018.

[13] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[16] B. Settles, M. Craven, and L. Friedland, "Active learning with real annotation costs," in *Proceedings of the NIPS workshop on cost-sensitive learning*. Vancouver, CA:, 2008.

[17] D. Yoo and I. S. Kweon, "Learning loss for active learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[18] J. Zhang, X. Wu, and V. S. Shengs, "Active learning with imbalanced multiple noisy labeling," *IEEE transactions on cybernetics*, 2014.

[19] T. Liu and D. Tao, "Classification with noisy labels by importance reweighting," *IEEE Transactions on pattern analysis and machine intelligence*, 2015.

[20] M. Fang, T. Zhou, J. Yin, Y. Wang, and D. Tao, "Data subset selection with imperfect multiple labels," *IEEE transactions on neural networks and learning systems*, 2018.

[21] A. A. Abdellatif, C. F. Chiasserini, and F. Malandrino, "Active learning-based classification in automated connected vehicles," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2020.

[22] D. A. Forsyth and J. Ponce, *Computer vision: a modern approach*. Pearson,, 2012.

[23] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan, "Image-based visual hulls," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 2000.

[24] Y. Yao and H. S. Park, "Multiview co-segmentation for wide baseline images using cross-view supervision," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020.

[25] C. Premebida, J. Carreira, J. Batista, and U. Nunes, "Pedestrian detection combining rgb and dense lidar data," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014.

[26] S. Gould, J. Rodgers, D. Cohen, G. Elidan, and D. Koller, "Multi-class segmentation with relative location prior," *International Journal of Computer Vision*, 2008.

[27] Z. Mai, R. Li, J. Jeong, D. Quispe, H. Kim, and S. Sanner, "Online continual learning in image classification: An empirical survey," *arXiv preprint arXiv:2101.10423*, 2021.

[28] Y.-C. Hsu, Y.-C. Liu, A. Ramasamy, and Z. Kira, "Re-evaluating continual learning scenarios: A categorization and case for strong baselines," *arXiv preprint arXiv:1810.12488*, 2018.

[29] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017.

[30] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.