APPLYING COLLABORATIVE ONLINE ACTIVE LEARNING IN VEHICULAR NETWORKS FOR FUTURE CONNECTED AND AUTONOMOUS VEHICLES

A Dissertation
Presented to
The Academic Faculty

By

Huiye Liu

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

August 2022

© Huiye Liu 2022

APPLYING COLLABORATIVE ONLINE ACTIVE LEARNING IN VEHICULAR NETWORKS FOR FUTURE CONNECTED AND AUTONOMOUS VEHICLES

Thesis committee:

Dr. Douglas M. Blough, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Linda M. Wills
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Raghupathy Sivakumar
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. May Dongmei Wang
Wallace H. Coulter Department of Biomedical Engineering
Georgia Institute of Technology

Dr. Yusun Chang
Department of Robotics and Mechatronics Engineering
Kennesaw State University

Date approved: April 30, 2022

How many ten years do one have?

– *Just do it :)*

To my family, for your endless love and support.

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere gratitude to my advisor Prof. Douglas M. Blough for the continuous support of my Ph.D study and related research, for his patience, motivation, and immense knowledge. I started my Ph.D. journey without a master degree and limited prior research experience, without his step-by-step help, I wouldn't be able to complete the related research at this moment. His guidance helped me in all the time of research and writing of this thesis. I could not have imagine having a better advisor and mentor for my Ph.D. study.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Yusun Chang, Prof. Raghupathy Sivakumar, Prof. Linda Wills, Prof. May Wang, for their insightful comments and encouragement, but also for the hard question which incented me to widen my research from various perspectives.

My sincere thanks also goes to Dr. Joyelle Harris. Thank you so much for offering the opportunity to let me serve as an instructor for ECE3710. It has been great support for my Ph.D. study and I gained invaluable teaching as well as communication experience, which will definitely benefit me in my future career path.

I am also pleased to say thank you to the Toyota Infotechnology Center, especially for Dr. Chung-Wei Lin, Dr. Eunsuk Kang, and my supervisor Shinichi Shiraishi. Without the internship opportunity and your kind support as well as guidance, I wouldn't be able to start such an amazing research journey in the path of connected and autonomous vehicles.

I would always remember my fellow labmates, Qiang, Mengyao, Hemin, Chuanji, Yuchen, Ang, Yan, Jingyuan and Lei, for the fun-time we spent together and insightful discussions on various research topics. I would also like to thank my dear friend Yundong Zhang from Stanford University, who spent many sleepless nights with me that gave me many insightful feedback and the courage to complete tasks before deadlines.

Last by not least important, I am grateful to my parents, friends and acquaintances who

remembered me in their prayers for the ultimate success. I consider myself nothing without them. They gave me enough moral support, encouragement and motivation to accomplish the personal goals. My two lifelines (parents) have always supported me financially so that I only pay attention to the studies and achieving my objective without any obstacle on the way. I could never successfully complete my Ph.D. program without your trust, sacrifice and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

AI artificial intelligence

AVs Autonomous Vehicles

BFCV Byzantine-fault-tolerant consensus algorithm

CAVs connected and autonomous vehicles

ML machine Learning

V2V Vehicle to Vehicle

SUMMARY

The main objective of this thesis is to provide a framework for, and proof of concept of, collaborative online active learning in vehicular networks. Another objective is to advance the state of the art in simulation-based evaluation and validation of connected intelligent vehicle applications.

With advancements in machine learning and artificial intelligence, connected autonomous vehicles (CAVs) have begun to migrate from laboratory development and testing conditions to driving on public roads. Their deployment in our environmental landscape offers potential for decreases in road accidents and traffic congestion, as well as improved mobility in overcrowded cities. Although common driving scenarios can be relatively easily solved with classic perception, path planning, and motion control methods, the remaining unsolved scenarios are corner cases in which traditional methods fail. These unsolved cases are the keys to deploying CAVs safely on the road, but they require an enormous amount of data collection and high-quality human annotation, which are very cost-ineffective considering the ever-changing real-world scenarios and highly diverse road/weather conditions. Additionally, evaluating and testing applications for CAVs in real testbeds are extremely expensive, as obvious failures like crashes tend to be rare events and can hardly be captured through predefined test scenarios. Therefore, realistic simulation tools with the benefit of lower cost as well as generating reproducible experiment results are needed to complement the real testbeds in validating applications for CAVs.

To address the above challenges, we propose a new collaborative distributed online active learning framework in vehicular networks supporting future CAVs. Four main works are summarized as follows. First, as the success of collaboration among vehicles heavily relies on the information exchange among vehicles, a novel byzantine fault-tolerant distributed consensus framework is proposed to secure critical information that is disseminated among nearby vehicles. Second, we present a novel framework for task-oriented

group formation, which allows computation tasks to be computed distributedly through collaboration among nearby vehicles to release the burden of offloading all data and computation tasks to remote clouds. Third, a new online active learning method is proposed to improve model accuracy without the support of human labelers as well as a remote centralized cloud, which allows groups of vehicles to cooperatively and actively learn with local data. Fourth, to enable the simulation study of our proposed collaborative distributed online active learning framework in vehicular networks and other CAVs algorithms/applications, we propose and build an integrated simulation environment, which integrates 3D scenarios, traffic/mobility, and inter-vehicle communication by synthesizing three existing open-sourced simulation tools.

CHAPTER 1
INTRODUCTION

With the rapid development of hardware devices such as LiDARs, depth-cameras and related technologies such as artificial intelligence (AI) as well as machine Learning (ML), it is now feasible for connected and autonomous vehicles (CAVs) to move to the next-level of achieving accurate localization, high-level path planning, behavior arbitration, and potentially supporting emerging applications such as real-time video streaming, virtual reality, and augmented reality in future CAVs models. However, current learning methods are mainly based on supervised and centralized training paradigms. Though such learning style has demonstrated its great potentials, its accuracy heavily relies on the quantity and quality of data sets and data annotation. Obviously, collecting such large data sets including all possible scenarios as well as corner cases along with high-quality human labels has many challenges. Therefore, new paradigms of learning styles such as online learning, active learning, etc., which can reduce the need of data collection/human annotation and improving learning efficiency are becoming a necessity [1, 2, 3].

1.1 Motivation and Research Objectives

The first priority of designing a vehicle must be safety, and there are no exceptions for CAVs. Therefore, as machine/deep learning become the backbone of CAVs system, higher detection and prediction accuracy are required than in other machine/deep learning intensive industries, e.g. internet service recommendation, advertisement, translator, etc. These systems are expected to operate flawlessly irrespective of different visibility, weather conditions, road surface quality, and even in the face of emergencies. In order to achieve the goal, large amount of data needs to be captured, transported from the vehicles to the remote data clouds, stored, analyzed, and properly processed then be able to used for training

models. As mentioned previously that supervised learning algorithms remain the dominant learning paradigm in the automotive industry, thus high-quality data annotation are also needed. Moreover, if a segmentation training task, unlike classification tasks, is needed, then segmenting (labeling an image pixel by pixel) pedestrians, cars, lanes, and other entities will become a significant bottleneck if the data annotation team is not adequately sized.

Therefore, it is clear that the effectiveness of current machine learning systems is directly tied to the availability and quality of training data. Having a large quantity of high-quality data will be necessary for algorithms to achieve a high accuracy. Nonetheless, the daily data recorded by an autonomous vehicle (typically equipped with 10 sensors/per vehicle as in KITTI standard [4]) can go up to the order of petabytes, posing challenges on the parallelization of the training procedure, as well as on the storage infrastructure, and not to say the annotation generation. As a matter of fact, vehicles can never encounter all complex traffic scenarios, where there are too many factors to enumerate and local storage is limited. More importantly, an autonomous vehicle should be able to adapt to different environments like leaving from suburban to urban area, change from sunny to rainy weather, etc. Thus, fast model updates and adaptability to new environments are also required. In other words, though, in general, more data is needed, it is of great significance to obtain a fuller coverage of different complex scenarios, instead of simple repetitive cases.

Considering the above limitations, an online active learning system is required before CAVs' great potentials can be realized. First, to release the burden of heavy data collection, learning should shift from offline to online. With high volume data, it is often infeasible to read-in/transfer all data at once. Since online learning can update its model using only the newest data points, the system does not need to store a large amount of data, which is ideal for vehicles with limited memory. Second, by processing only a small chunk of data at a time, such learning method keeps the computational complexity of each update small. It makes it feasible for a vehicle to update its model distributedly without a remote server,

thus saving the channel bandwidth for other needed applications. Third, as active learning allows a vehicle to choose the data from which it learns, a vehicle will only need to learn a piece of data which it is not capable of doing, and saving both computational and storage resources. Moreover, online active learning can gradually discount the importance of past data without a supervisor, where the model automatically adapts to changes in the dataset, which is especially useful for dealing with changes to the environment encountered by a vehicle.

However, despite the conceptual appeal, several challenges are posed in the vehicular network environment. First, there are not labelers in a vehicular network that can generate valid labels for data captured at run time, where collaboration among vehicles becomes inevitable, as a single vehicle with limited view and information can hardly obtain qualified labels. Second, vehicles are moving fast, where the network topology is transient and unreliable, making the cooperation among vehicles very challenging. Though online learning is designed for fast convergence, compared with the ever-changing vehicular topology, it is not guaranteed that neighboring vehicles can stay together long enough before their cooperative tasks are completed. Third, as information and data sharing are unavoidable, critical data reliability and verification of actual proximity to target event is extremely important to training quality as online models can become unstable or corrupted due to contaminated data.

Therefore, a collaborative online active learning framework is needed. Our focus of this work is on the framework design and proof-of-concept analysis of the realization of this learning paradigm in the vehicular network environment. The objective of this thesis is to address the challenges therein and establish the fundamentals of the collaborative online active learning framework in vehicular network for future connected and autonomous vehicles.

1.2 Contributions

The primary contributions of this thesis are summarized as follows:

- The first contribution (Chapter 3) is that we propose the concept of Proof-of-Eligibility Challenge, which proves vehicles' actual presence to a claimed event such as an accident at an intersection, and we designed a Byzantine-fault-tolerant consensus algorithm for connected vehicles (BFCV) to ensure validity of event reports disseminated in the network, without requiring privileged members, leader election, nor trusted shared key distribution. Consider an example scenario where a vehicle is trying to improve its accident detection through captured images and is collecting data to improve its model accuracy. In a vehicular network without critical information validation scheme like BFCV, a vehicle might believe another vehicle's false warning about an occurrence of an accident, falsely consider that its captured data has indicators of an accident, and update its accident model accordingly. This will not only add fluctuations to existing model, but also affect model accuracy negatively. To verify the effectiveness of this approach, we implemented a BFCV prototype and simulated it in a realistic environment built on top of Veins, SUMO and OMNet++. Evaluation results show that BFCV provides fast and secure consensus satisfying both safety and liveness requirements.

- The second contribution (Chapter 4) is a framework to address task-oriented group formation, based on the probability of successful task completion. Once the reliability of disseminated information can be verified, a good task-oriented group formation scheme among nearby vehicles is critical for enabling vehicles' capability for online active learning. It not only eliminates the high latency caused by communicating with remote servers, but also improves quality of service as aggregating nearby vehicles' diverse views can actually improve the results of the learning tasks. We first discovered that the probability of cooperative task completion is primarily dependent

on two random variables: 1) stay time – the length of time that a group of vehicles that are cooperating stay in communication range of each other; and 2) task completion time – the length of time required for the task to be completed by the group of vehicles. In order for a task to be successfully completed by a group, the task completion time should be shorter than the stay time. However, in practice, it is not possible to know the exact probability distributions of the stay time and task completion time of a group. Thus, we proposed and evaluated a heuristic two-stage algorithm that performs task-oriented group formation for cooperative computations. The algorithm maximizes the size of groups in order to produce the best cooperative result while ensuring a specified probability of task completion. Extensive evaluation results show that our algorithm achieves high task completion rates and good group sizes across a wide range of traffic scenarios.

- The third contribution (Chapter 5) is that we propose a collaborative online active framework realizing the benefits discussed above. To replace human labelers, we present 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. To facilitate label generation and improve label accuracy, a multi-view prediction transfer scheme is designed to align different views from multiple vehicles via leveraging sensor data fusion. Besides, 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 is proposed. An implementation of the framework is provided and extensive evaluation are conducted to demonstrate the effectiveness of the approach.

- The fourth contribution (Chapter 6) is that we present a co-simulation framework

(Sim2Scale) to develop, simulate, and evaluate algorithms and methodologies for collaborative computation tasks, which enables simulation of traffic, communication and 3D environment in connected and autonomous vehicles. Both Python script and QT GUI interfaces are provided for users with less familiarity with existing simulators to quickly and easily build and validate their ideas. Moreover, a collaborative active learning example application is presented, which can be extended to other cooperative or non-cooperative tasks with modest effort. [1]

1.3 Organization of the Thesis

The rest of the thesis is organized as follows. In Chapter 2, we provide a brief introduction on the connected and autonomous vehicle environment, the information security in the vehicular network, review of the state of the art machine learning as well as AI technology, and corresponding test and validation tools. Then, in Chapter 3, a novel byzantine fault-tolerant distributed consensus framework is proposed to secure critical information exchanged among vehicles. In Chapter 4, the problem of forming a task-oriented group which can achieve both quality and quantity of task completion is addressed. A new online active learning method is presented in Chapter 5 to ease the burden of human labeler and reliance of remote data centers. Moreover, in Chapter 6, we propose and build an integrated simulation tool, which integrates 3D scenarios, traffic/mobility, and inter-vehicle communication by synthesizing three existing open-sourced simulators. Finally, in Chapter 7, our conclusion and future works are discussed.

[1] This work, *Sim2Scale: A Co-Simulation Framework for Developing and Evaluating Algorithms for Connected and Autonomous Vehicles,* Georgia Tech Technical Report (to be submitted as a conference publication. The entire Sim2Scale framework and the collaborative active learning example application will be open-source and freely available to the research community after paper submission.).

CHAPTER 2

BACKGROUND AND PRELIMINARIES

Connected and autonomous vehicles (CAVs), namely, are the combination of connectivity and automated technologies to assist or replace humans in the task of driving. This can be realized through a combination of advanced sensor technology, on-board or remote computing capabilities, GPS and telecommunications systems, etc.. Therefore, in this chapter, we will introduce the preliminary knowledge of the connectivity of vehicles – vehicular network, review the current status of the automated technologies, especially in artificial intelligence and machine learning, and the test as well as validation of the CAVs system.

2.1 Overview of Vehicular Networks

The concept of "Connected Vehicle (CV)", was first brought up in 1996 by GM working with Motorola to initiate voice calls to emergency responders in the case of accidents when airbags were deployed. Though without a formal definition, "Connected Vehicle System" (as shown in Figure 2.1) nowadays is a network of smart vehicles and infrastructure/edge devices (for example RSUs), which are equipped with powerful multi-functional sensor sets, processors, memory and wireless communication devices, communicating with each other to warn of safety alerts, provide smarter navigation, improve transportation efficiency, etc. To date, various communication technologies and smart applications have been studied to realize the "fantasy" on roads. However, as the benefits grow, new challenges, e.g., information security, also emerge. Therefore, in this section, we will go through the preliminaries and some research results to provide an overview of the vehicular network environment.

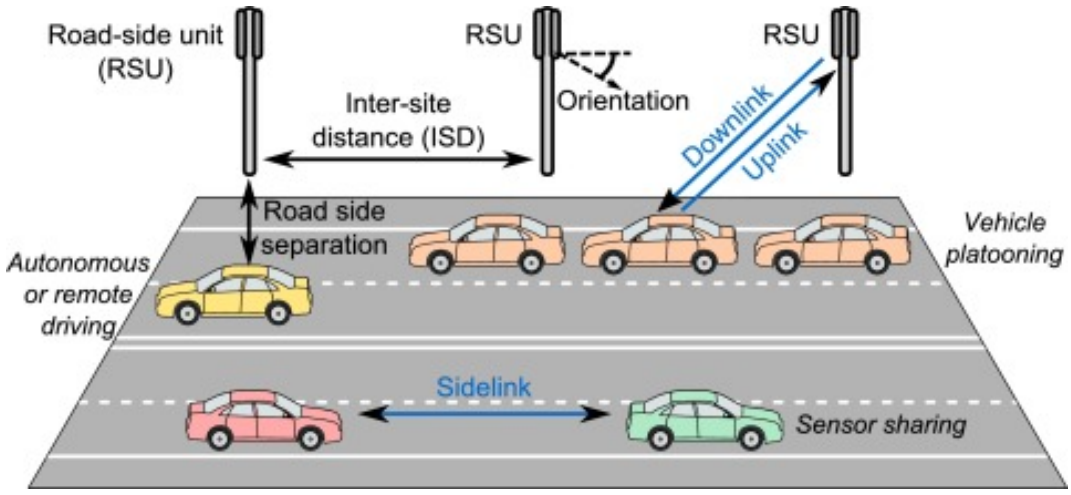


Figure 2.1: Connected vehicle system and emerging vehicular applications [5]

2.1.1 The Connectivity of Vehicles

One possible way of achieving inter-vehicle communication is through Dedicated Short-Range Communications (DSRC), for which The Department of Transportation (DOT) is paving its way to deployment [6, 7]. DSRC is designed for Vehicle to Vehicle (V2V) and Vehicle to Infrastructure (V2I) communication, which allows a vehicle to broadcast its position, speed, acceleration, moving direction, etc. in Basic Safety Message (BSM). It also supports emergency messages like collision or congestion report through multi-hop broadcast. Besides the V2V/V2I communication, DSRC can also be applied to road-side units (RSUs), pedestrians, bicycles, and other devices with a wireless capability. Interchangeably, the combination of V2V and V2I is referred to as vehicle-to-everything (V2X) or wireless access in vehicular environments (WAVE) for the IEEE portion. The connectivity provides appealing advantages to various applications such as emergency warning, cooperative adaptive cruise control, and intersection management without traffic lights.

This technology has been standardized and field-tested by major car companies [6, 7]. Announcements [8] have been made about deploying DSRC in US market from 2021 by Toyota and in Europe starting from 2019 by Volkswagen. General Motor [9] has already introduced DSRC in Cadillac back in 2017, which can handle 1000 messages per second

from vehicles up to nearly 1000 feet away.

Cellular Vehicle to Everything (C-V2X) [10], developed in the 3rd Generation Partnership Project (3GPP), is another connected vehicle technology considered as an alternative to DSRC. Though no C-V2X products are available today and are far from reaching the maturity and security levels needed for critical safety systems, Ford announced plans to adopt C-V2X early this year. C-V2X appears to be more ambitious and have bigger potentials that supports not only direct communications (V2V, V2I, V2P, etc.) that DSRC adopts, but also supports network communications, in which C-V2X employs the conventional mobile network to enable the vehicle to receive information about road conditions and traffic in the area, but it is far more costly and complex, which rooted from base-station implementation.

Besides, as the emerging 5G millimeter-wave (mmWave) technology can provide rich spectrum resources to support the timely transmission of large amounts of data, it attracts recent research attentions in vehicular networks. Consider current smart vehicles are equipped with large number of sensors generating huge data set in fly, expanding the benefit of mmWave to vehicle-to-vehicle propagation channels is of great importance. Though discussion and research works as in [11, 12] are appearing, it is still at a very early stage of exploration.

In summary, it is still at an early stage of deciding DSRC and C-V2X, or even other new technologies, connected Vehicles is an unavoidable trend attracting significant attention and can drastically reduce road fatalities, improve traffic efficiency, and enable vehicle automation. It is important for related research work to not limit with only one communication technology.

2.1.2 The Need for Group Formation in Vehicular Networks

Thanks to the advancing communication technology in the vehicular network, immersive emerging applications such as advanced driver assistants, safety improvement, intelligent traffic lights, and autonomous driving that are promising to improve the road safety and

enhance the traffic efficiency. However, these applications have heavy computational requirements [13] and cannot be offloaded to remote clouds due to delay and bandwidth constraints [14]. The computational requirements of many potentially useful applications of this type also exceed the capacity of individual vehicles [15]. Thus, computational task offloading at the edge is essential for intelligent networked vehicles to reach their full potential. The literature in mobile edge computing (MEC) [16, 17, 18], vehicular fog computing (VFC) [13, 19, 20, 21], and vehicular cloud computing (VCC) [14, 22] has widely studied the problem of task offloading including aspects such as task scheduling [23], resource allocation [24], and computation complexity [25]. However, this prior work focuses primarily on one-to-one task offloading and also relies on infrastructure support in the form of edge servers and/or roadside units. One-to-many task assignment without infrastructure support is neglected.

Clustering algorithms provide one possible approach to forming vehicular groups for one-to-many task offloading. Clustering work in dynamic networks began with studies on MANETs [26, 27]. Various clustering design goals, e.g. load balancing, cost of clustering, speed of cluster formation, and real-time requirement [26] were achieved through single or combined metrics of connectivity, mobility, power, maintenance cost, etc. [27]. As discussed in [28], MANETs and VANETs are different in many ways, especially in mobility patterns, network topology, and communication link lifetime. With a goal of maximizing stability, clustering studies in VANET have considered mobility more thoroughly [29, 30]. However, these works do not factor in the characteristics of the tasks to be processed when determining a good group of vehicles to cluster. Specifically, the goal of forming a group is to complete certain tasks. Even if a formed group has the best stability, but not able to complete the task well, then the formed group becomes useless and a waste of resource. However, if other types of groups are formed, though in a less stable way, and it can complete the task well, then it still fulfill the goal of forming the group. Therefore, it is important to design group formation schemes based on the task goal.

Besides, in addressing task offloading for many cooperative computation applications in vehicles, where the quality of the results increases with the number of participating vehicles [31, 2], there is an interesting interplay between result quality and successful task completion. This is because larger groups improve result quality but also reduce the group stay time, which makes it less likely that the cooperative computation will complete while the task group remains together. Therefore, neglecting the important relationship between group stay time and task completion time could result in low task completion rate.

In summary, it is important for future clustering or group formation works to take task characteristics, task completion quality and task completion rate into consideration and make proper trade offs based on the design goals.

2.1.3 Information Security in Vehicular Networks

Each coin has two sides. As we are benefited by the smarter and more connected vehicular network, the security of the flowing information and who to trust in the network becomes a new challenge. Thus, in this section, we first review the existing approaches proposed for information security in connected vehicles, then followed by a short discussion.

Node centric methods such as reputation systems [32, 33, 34, 35] are popular and have been widely studied in the past. Such system inspect the past and present behavior of nodes and use this to predict the future misbehavior. All methods falls into this category hold a same assumption, such that the nodes who behave well in the past are more likely to behave well in the future. However, smart adversaries may only initiate attacks at critical times, which is a fundamental problem that reputation systems cannot handle.

As discussed in [36], data centric methods focus on analyzing transmitted data among nodes and information verification. Vehicles that use local methods, e.g. [35, 37, 38], verify information locally without relying on other vehicles' cooperation. Though these methods are light-weight, easy to scale and can tolerate intermittent communication, they heavily rely on location information and have a limited view of what is happening on the

road, which reduces their accuracy.

Other cooperative schemes, as surveyed in [36], are more accurate than local methods with lower false positive and false negative rates. Nevertheless, they are more vulnerable to packet loss/delay and the ratio of compromised to honest vehicles. Threshold-based voting has been adopted in PoR [39], DC [40] to filter false data that honest vehicles only accept a report when they receive more than X signatures attesting to it. PoR improves the efficiency of communication by using growth codes, but the performance is sensitive to a preset threshold, while the DC method in [40] dynamically sets the threshold based on the criticality and the density of the network. However, [40] assumes 1-D communication, detection, and reaction, primarily limiting its use to highway scenarios. Moreover, threshold voting does not provide true consensus, because each vehicle decides independently and, therefore, different honest vehicles can reach different decisions.

Consensus can be another alternative path to secure information security in the vehicular network. Broadly speaking, consensus is a process to reach agreement on data values or decisions among a group of participants. Achieving consensus in a distributed system is essential but challenging as a consensus algorithm should be resilient to node failures, network disconnections, communication delays, and even malicious attacks. In general, consensus algorithms leverage building blocks such as atomic commitment, group membership, and total order broadcast. Any node in the network can propose a value or an opinion to the members of the consensus group. After some rounds of communication, all non-faulty and non-compromised group members agree on a single consensus value.

True consensus algorithms satisfying *termination, agreement,* and *validity* properties have been well studied in other contexts. Paxos[41] and Raft [42] are well known algorithms to achieve consensus among unreliable nodes, but they do not address Byzantine faults [43]. Moreover, they either require fixed network or are computationally expensive, which make them impossible to be applied to connected vehicle system. Similarly, other traditional byzantine agreement protocols, e.g. [44, 45, 46] , though address byzantine

faults well, do not handle highly dynamic network connectivity such as occurs in vehicular networks. In general, traditional consensus algorithms require heavy computation, frequent message exchanges, and/or a fixed wired network, making them hard to be adapted to vehicular networks. While efforts have been made to adapt traditional consensus algorithms for dynamic and intermittent topologies in MANETs, e.g. [47, 48, 49, 50], none of these efforts address both unreliable links and Byzantine faults.

The heated Blockchain (BC) technology, evolving to be widely used in Peer-to-Peer (P2P) network which are similar to connected vehicles' topology to provide security and privacy, leverages a wide range of consensus models, such as Proof of Work (PoW) [51] used in Bitcoin/Homestead Ethereum or Proof of Stake (PoS) [52] in Ethereum (Serenity). Nonetheless, PoW achieves security in an untrusted environment but causes huge computation overhead. Though studies [53, 54, 55, 56, 57] has been made to adapt BC technology to Internet of Things (IoT) system, yet the assumed model still partially relies on centralized cloud server and assumes no mobility of the nodes, thus making it not suitable for vehicular network.

In short, although information security in vehicular networks has been previously studied, efficient collaborative methods that guarantee all healthy nodes make the same decision have not been developed to date. Moreover, how to make sure that vehicles in the network will believe opinions from only safely selected honest vehicles with enough knowledge of the event is significant to filter out false information. To be specific, most prior approaches allow each individual vehicle to make its own decision without achieving true consensus among vehicles and cannot guarantee that vehicles in the network only accept evaluation decisions from honest vehicle. It is important for future works to design consensus-based measures that can achieve timely and efficient true consensus while supporting vehicles' high mobility as well as intermittent connections.

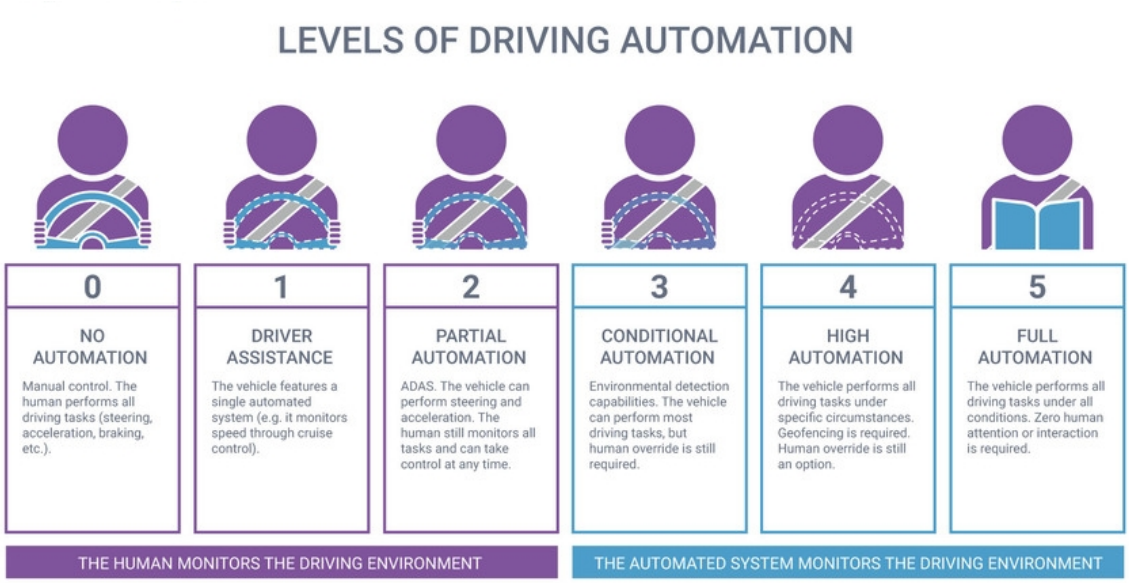


Figure 2.2: Level of driving automation [58]

2.2 Overview of Autonomous Vehicles

An autonomous vehicle, or self-driving car, is a vehicle which utilises a fully automated driving system in order to allow the vehicle to respond to external conditions that a human driver would manage. There are several types of autonomous vehicles, depending on their level of automation. The Society of Automotive Engineers (SAE) defines 6 levels of driving automation ranging from 0 (fully manual) to 5 (fully autonomous). These levels have been adopted by the U.S. Department of Transportation, as illustrated in Figure 2.2.

Most vehicles on the road today are from Level 0: manually controlled to Level 3: conditional automation, depends on the make and price. They still relies on human driver to provide the ”dynamic driving” decision making, although there may be smart systems like advanced driver assistance system (ADAS) and autopilot system in place to help the driver. Such systems are only for assistance, and, technically, they ”do not drive”. However, in the research works or media, when we say autonomous vehicles, we are referring to Level 5: full automation vehicles, which do not require human attention, where the “dynamic

driving task" is eliminated from human driver. Level 5 cars will be the true self-driving car, which will be free from geo-fencing and able to go anywhere and do anything like experienced human drivers. Though such fully automated cars are undergoing testing in several test beds around the world, but none are yet available to the general public. We are still at a very early research stage to level 5 cars.

Autonomous vehicles relies on two key components: 1) powerful sensor set, which enables the vehicle to sense and perceive the world like a human driver; and 2) advanced artificial intelligence (AI) and machine learning (ML) systems, which serve as the brain enabling vehicles' capability to understand what the sensor data means and how to react like a human driver. Therefore, in this section, we first provide an overview about complex sensors that a typical autonomous vehicle employs, then follow by a thorough literature survey of machine/deep learning algorithms for autonomous vehicles. Note that we use the terms artificial intelligence, machine learning, and deep learning interchangeably in the following discussion. We consider that artificial intelligence applies machine learning, deep learning and other techniques to solve actual problems.

2.2.1 How Autonomous Vehicles Sense the World

Autonomous vehicles would not live without sensors. They relies on complex sensor sets to "see" and sense what is on the road, as well as to collect needed information needed in order to make steering/driving decisions. The majority of today's automotive manufacturers most commonly use the following three types of sensors in autonomous vehicles: cameras, radars, and lidars, where

- ***Camera***: as the mostly commonly seen device in daily life, various types of cameras, e.g., video cameras, depth cameras, are install on autonomous vehicles in order to see and interpret the objects just like human drivers do with their eyes. By equipping cars with cameras at every angle, the vehicles are capable of maintaining a 360 degree view of the external environment, thus obtaining a holistic picture of the sur-

rounding traffic conditions. By embedding advanced machine learning algorithms, cameras is capable of not only capturing highly detailed and realistic images, but also automatically detecting and recognizing objects. For example, these cameras can capture/identify pedestrians, neighboring vehicles, buildings, traffic signs, lane markings, etc. as they move. Nonetheless, these camera are still far from being sufficient. Poor weather conditions such as fog, rain or snow can prevent cameras from clearly capturing the obstacles on the road surface, which increases the likelihood of accidents. Besides, there are often situations where the image resolution from the equipped cameras simply aren't good enough for a algorithms to make a good decision about what the car should do. For instance, in situations where the colors of objects are very similar to the background, a suspected pedestrian is only captured by limited amount of pixels, or the contrast between them is low, the machine learning algorithms can fail. Currently, Tesla's autopilot system is build based on camera systems.

- ***Radar***: with the full name as radio detection and ranging sensors, which sends out radio waves that detect objects and gauge their distance as well as velocity in relation to the vehicle in real time. Typically, both short- (24 GHz) and long-range (77 GHz) radars are installed on the an autonomous vehicle and each of them has their own functions. In general, short range radar is used to conduct blind spot monitoring, provide the ideal lane-keeping assistance and parking aids, while the long range radar sensors are used more for brake assistance and distance control. Unlike cameras, as discussed above, radar is not affected by light and darkness and with the ability to detect obstructions like glass. However, by using radars, today's autonomous vehicles can only correctly identify between 90% and 95% of pedestrians, which is hardly enough to ensure safety on the road [59]. Moreover, radars cannot size an object's height information, since they can only scan horizontally. This will become a problem when driving under bridges or road signs.

- ***Lidar***: with the full name as light detection and ranging sensor, works similar to radar sensors. The only difference is that instead of using radio waves, lidars use lasers . Apart from measuring the distances to various objects on the road, lidar allows capturing 3D point cloud of the detected objects and mapping the surroundings. Moreover, lidar can provide a much broader view of the surrounding environment, for example, a full 360 degree map. In light of the advantages, various autonomous vehicle provider, e.g., Toyota, Google, Uber, chose to apply lidar system in their design. However, lidars are much more expensive than radars and cameras. It has been estimated that the a lidar system required for an autonomous car can cost well beyond $10,000, while the top sensor being used by Google and Uber costs up to $80,000 [59], which is almost impossible for general use. Additionally, even with the best lidar system, poor weather like snow and fog may still negatively affect the system's ability to correctly detect objects on road.

Therefore, as there are no one-fits-all sensors, there haven't been a standard established of what sensors to use and how many sensors should be used. Nevertheless, KITTI sensor setup is a commonly used plan in research works. As proposed in KITTI standard [4], a combination of different sensors are used. 10 sensors are installed on top of a vehicle, including 1 inertial navigation system (GPS/IMU), 1 lidar, 2 gray-scale cameras, 2 color cameras, and 4 vari-focal lenses. The setup is shown in Figure 2.3, and a real life example setup on a Volkswagen Passat B6 is shown in Figure 2.4.

Moreover, such sensor set is only build for a vehicle to sense the world. Other sensors/equipment are needed for an autonomous car to communicate with the world, e.g., nearby vehicles, road-side units, smart traffic light, remote data centers, and so on. By collecting information with a vehicle's own sensor set and exchanging the collected information through V2V, V2I, V2X communication (refer to Section 2.1.1 for details) with the world, the vehicle is able to sense the actual path ahead, traffic jams, obstacles on the road surface, pedestrians, etc..

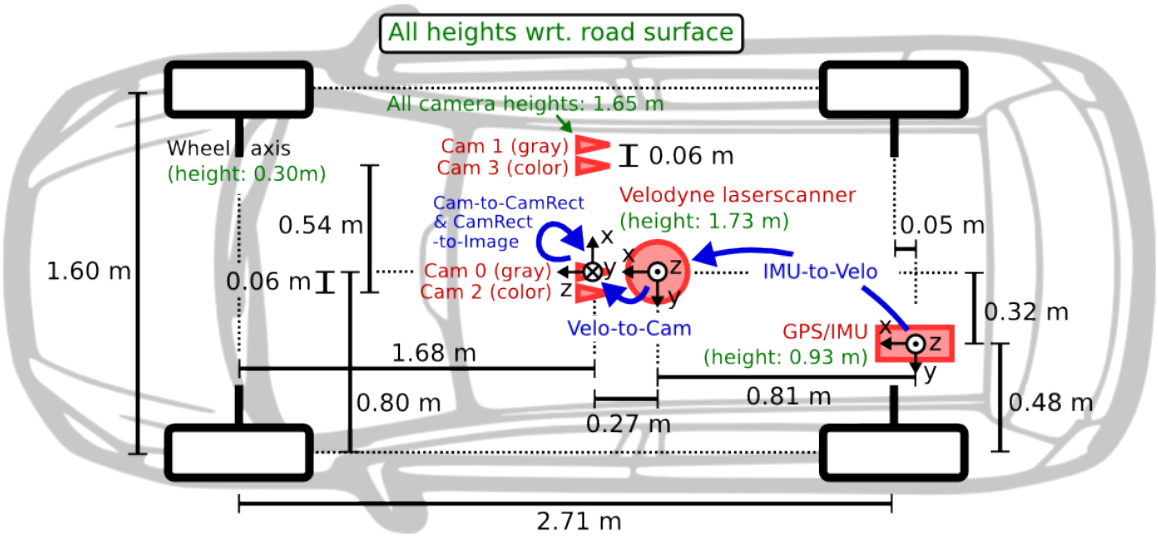


Figure 2.3: KITTI sensors setup of an autonomous vehicles [4]

2.2.2 Machine Learning: the Backbone of Autonomous Vehicles

Once an autonomous car can correctly sense the world, how to process the collected information and make proper decision based on it becomes a challenge. In the past 10 years, the advancement of machine learning has truly stimulated the development and deployment of autonomous vehicles in the transportation industry. Fueled by big data from various sensing devices and advanced computing resources, machine learning has become the backbone of AVs for perceiving the surrounding environment and making appropriate decision. To achieve goal of full automation (i.e., level 5 self-driving), extensive research efforts have been made in the field of machine learning and deep learning algorithms to making the vehicle smarter.

This section surveys the research works on advanced learning techniques to optimize learning based applications in autonomous vehicles, which directly affects the topic of this thesis work. We first give a short overview of machine learning technologies in vehicular network, and summarize the standardized works in related advanced learning methodologies. Then we review the research on general vehicular network utilizing online and active

Figure 2.4: Connected vehicle system and emerging vehicular applications [4]

learning technologies.

The Advances of Machine Learning

Machine learning, which optimize performance by directly learning on the samples from the targeted environment, has been widely used to solve conventionally challenging problems in automotive industry such as [60, 61, 62, 63, 64, 65], showing its strong potentials in precision and adaptation. More recently, deep learning, as a sub-set of machine learning, has dominated the model design in many engineering applications. In light of its appealing capability of self-picking features and supreme accuracy achieved when trained with large amount of data, recent vehicular applications are taking advantages of deep learning to boost the performance as well [66, 67, 68, 63, 64].

Machine learning techniques, in general, is applied to vehicular applications in two major ways: 1) *for vehicle purpose* - applications includes scene perception, localization, path planning, behavior arbitration, etc. are all used to improve the driving capability of an ego vehicle [3]; 2) *for infrastructure purpose* - applications such as traffic flow prediction, network congestion control, load balancing, etc. are investigated to provide better vehicular network connectivity and efficiency as surveyed in [1].

Nevertheless, it is not a panacea to all challenges. Naively applying existing machine-learning methods to vehicles is expected to be insufficient due to their distinguishing characteristics. First, conventional machine learning are done in batch mode, where a model is trained with the entire data set at once then sent out to user for inference; however, many real-wold scenarios are not like so. For instances, a vehicle's trip time is measured when it departs from source and until arrives at destination. The duration can affected by weather, traffic condition, driving habits, etc. [69]. It is impractical to obtain a full data collection and a one-fit-all super model.

Second, as most existing vehicular applications are supervised-based machine learning, large data collection and high quality human annotations are taken as standard procedures.

Nonetheless, as reported by [3], daily data recorded by an autonomous car can reach the order of petabytes, not to mention the effort required for data labeling, storing and cleaning. Accordingly, the real challenge hindering the development of machine learning in vehicular network lies in the availability of training data and real-time computing constraints.

Thus, to meet the ever-increasing training data demand of various AV applications, both academia and industry are pursuing new learning technologies beyond conventional paradigms. Online learning [70] and active learning [71] have attracted big attention due to its capability of training with sequential data and reduced label requirement, which is considered as key enabling technologies in various fields such as speech recognition [72], security [73], and we believe it will also bring significant benefits to AVs. In what follows, the related works on advanced learning techniques are discussed, which involves online learning and active learning.

Online Learning

As opposed to batch (offline) learning, online learning allows incremental model updates with streaming data, which largely reduces re-training cost and adapts to new data-points quickly as well as efficiently. Besides, online learning trains as data comes, which alleviates the tension for data storage. Given these appealing nature, it has been widely deployed on various machine learning based applications such as spam filtering [73], travel time prediction [69], personalized recommendation systems [74], and etc.. As summarized in [75], online learning techniques are often used for two purposes: 1) handle learning from large-scale streaming data, where it is infeasible to train over the entire dataset, and, situations where it is necessary for the model to dynamically adapt to new patterns in the dataset, like travel time prediction of vehicles where data arrives periodically and the learner has to adapt to new datapoints immediately before getting the next round [69]; 2) improve efficiency and scalability of existing machine learning methodologies for existing batch machine learning tasks where a whole dataset must be made available before the learning

tasks, such as SVM algorithms [70].

However, though data in vehicular network usually comes sequentially and in large-scale, where online learning appears to be a magic cure, yet, it remains a non-trivial task due to the high volume and high velocity of vehicular network. Considering the data need to be collected and processed scales from a single vehicle to vehicles moving on a highway such as I-75 N in Atlanta, the order of the daily data may grow from 10^{15} bytes to 10^{21} bytes[1]. In this way, even newest 5G technology may not be able to support all vehicles offloading learning tasks to servers, and local data processing cannot be avoided. Therefore, local collaborative online training and high performance computing should be investigated to tackle these challenges [75].

Active Learning

With the growing demand of learning with less labels, active learning, which can achieve greater accuracy with fewer labeled training data is drawing attention in various fields such as speech recognition [76], image processing [68], vehicle detection [77], malware detection [78], etc.. In active learning, the algorithm can interact with an oracle (e.g., a human annotator, super-accurate existing models) by querying the label of a selected datapoint. The labeled selected data are then treated as a "representative", where the model will converge much faster on it, requiring less efforts of annotation. For instance, high quality labels of speech utterances is extremely time consuming and hard to obtain, which requires trained linguists to spend, e.g., about 400 times longer than the actual audio length to label phonemes [79]. With an active learning approach, [76] reduce the requiring labels by 6 times while maintaining similar accuracy.

The main process of active learning is called query, where an representative data instance is selected and asked for label. As surveyed by [80], active learning can be divided into three settings, based on the query strategies: 1) membership-based, where a learner can

[1]The estimated count of vehicle pass through I75N stations is inferred based on annual average daily traffic (AADT) recorded in 2019 (https://gdottrafficdata.drakewell.com/publicmultinodemap.asp)

ask labels for any data instances; 2) stream-based, where data instances are sequentially sampled and the learner decides whether to query the sampled instance based on certain measures such as thresholding; and, 3) pool-based, which is similar to stream-based approach but it evaluates and decides query items once based on the entire collection instead of one by one. Pool-based setting is widely used as in [76, 77, 68, 78], because it allows learners to select the best suited queries given the knowledge of all samples. However, stream based approach is more appropriate for distributed learners with limited memory and computing capability like vehicles.

Given active learning aims at finding the minimum amount of labeled data to achieve a certain performance, it has been considered as a promising approach to address the data and efficiency challenges in vehicular network. However, sitting a human oracle to label sampled data largely hinders practical deployment. Replacing oracle-based labeling with other noisy label generation methods are anticipated to thrive for future autonomous vehicles.

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 [2]. 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. Moreover, data freshness shows weaker indicator for image-based tasks. Other unique metrics in vehicular networks should be considered. Li, et al. take advantage of the multi-view effect to address the partial occlusion issue in vehicle detection [77]. Nevertheless, their set up is specific to the detection problem and is not easily adapted for other tasks like the segmentation task, and their approach also assumed human annotators for label generation.

Online Active Learning for Vehicular Networks

As introduced in Section 2.2.1, autonomous vehicles are equipping with powerful sensors, e.g. LiDARs, radars with different ranges, camera, ultrasonic sensors, speed sensors, etc. Although, these provides wealthy, real-time, and easily attainable data sources for machine learning, generating valid labels for such large scale data is very expensive. As high performance embedded processors are landing in modern vehicles [81], active learning can be leveraged to learn from these data with much reduced labeling effort. Previous works in vehicular network, e.g. [77, 82, 83, 84, 85] uses active learning to train models with less labeling costs. For example, [83] show that human oracle with selective sampling can improve vehicle recognition accuracy with reduced labels, and with manually selected query [85] is able to identify vehicles from images in the face of partial occlusions. However, human oracle is required for above works, which is rather unrealistic in real vehicular network. Furthermore, as summarized in Section 2.2.2, re-training is very expensive and unrealistic when data comes in sequence.

Other works, e.g. [86, 87, 2] is able to actively learn with sequential data by combining online learning and active learning techniques. For instance, [86] proposed an online active learning algorithm for frame-subset selection problem in driving videos to increase the processing efficiency. By using a combination of multiple similarity criteria, the algorithm actively maintain a set of key frames from either streaming or batch data and observe 100:4 video compression rate while not affecting the performance of downstream tasks. Gaussian process regression approach is used in [87] to actively learn the vehicular communication network dynamics (i.e. wireless channels and interference) in real-time, which significantly improved the transmission reliability and minimized the age of information violation probability—a key metric in ultra-reliable low-latency vehicle-to-vehicle communication. Besides, this is a decentralized method, which adapts to the changing environment very fast. Abdellatif et al. in [2] also proposed an online active learning framework to improve the accuracy of vehicle classification, where no human oracle is used. It takes advantages

of the V2V communication by asking neighboring vehicles to cooperatively decide pseudo labels for captured data. Nonetheless, naive majority voting is used for deciding pseudo labels, without considering other factors – initial model accuracy, view angle to the target, etc. Besides, it ignores the importance of communication stability with neighboring vehicles, where participating vehicles may lose connections before the learning process completes.

2.3 Evaluation and Validation of Applications for Connected and Autonomous Vehicles

As mentioned at the beginning of this chapter, CAVs are indeed the combination connectivity and automation. Without either component, the CAVs will not be powerful enough to realize the level 5 fully automated goal. Despite the research advanced as discussed in Section 2.1 and Section 2.2, how to properly evaluated the two main components of CAVs is of the same importance as algorithm development. Considering both the module development demand by car makers and safety requirement established by policymakers, the CAVs' communication module as well as automation functions must be extensively evaluated both individually and synthetically prior to deployment. However, the evaluation process for CAVs is challenging because obvious failures like crashes tend to be rare events and CAVs with problems can slip through by passing predefined test scenarios. Thus, how to identify problems and failures that will manifest in the real world becomes the main goal for evaluation and validation approaches.

2.3.1 Test on Roads

One evaluation approach is based on naturalistic field operational tests, where prototype CAVs are driven on roads by volunteers or test engineers [88]. Common driving scenarios can be reproduced with thousands of miles driving on typical roads or real world test beds such as Metropolitan Transportation Commission (MTC) [89] and Ann Arbor Connected

Vehicle Test Environment (AACVTE) [90]. However, uncommon scenarios, e.g., weather quickly changing from sunny to heavy rain or a sudden car collision 30 meters ahead, are corner cases in which prototype CAVs may fail. These corner cases are the keys to safely deploying CAVs in the real world but, unfortunately, cannot be easily reproduced through the naturalistic field operational test approach.

2.3.2 Test through Simulators

To complement the naturalistic field test approach and reduce cost, realistic simulations are essential. Existing simulators can be divided into three main categories based on the evaluation purpose: 1) traffic and vehicle mobility, where either or both microscopic/macroscopic vehicle mobility models are adopted to simulate different traffic scenarios in a given area map [91, 92, 93, 94]; 2) vehicular network communication, in which V2X communication is simulated in detail accounting for aspects such as multi-channel operation and noise/interference effects [95, 93, 96, 94, 97]; and 3) 3D environment with sensors, where realistic landscape (including buildings, pedestrians, roads, etc.) is modeled to support the holistic simulation of sensors used by CAVs, e.g. lidar, camera, and radar [98].

Nonetheless, though the existing simulators demonstrate superior performance in fulfilling their individual design goals, no open-source simulation platform exists that combines all three categories at the same time, which is required for many of the CAV applications discussed in at the beginning of Section 2.3. While InfoRich [99] presents a co-simulation platform targeting vehicle energy consumption and includes simulation of vehicle dynamics, sensors, and V2X communications, it is not based on fully open-source libraries and is not well suited for adaptation to non-energy-related algorithm evaluations.

2.4 Chapter Summary

In this chapter, the basic background and preliminary knowledge related to connected and autonomous vehicles are provided. The first section focuses on the overview of vehicular

network, including the communication technologies and some of the important research aspects of the connected environment. Second, we provide a general background about autonomous vehicles, especially for its sensor sets, then survey the important related research works of machine learning, which drives the thrive of vehicle automation. Last but not least important, a short discussion of test and validation of applications for CAVs is provided to complete the general background of connected and autonomous vehicles.

CHAPTER 3
BFCV: BYZANTINE-TOLERANT DISTRIBUTED CONSENSUS FOR CONNECTED VEHICLES

The growing connectivity and emerging applications in connected vehicles have tremendous potential for advances in safety, navigation, traffic management and fuel efficiency, while also posing new security challenges such as false information attacks. Without securing the information exchanged in the network, applications relies on messages sharing such as Advanced Driving Assistance System (ADAS), Cooperative Intelligent Transport Systems (C-ITS) may not be able to work, or even cause accidents. As a result, this chapter targets the problem of securing critical information that is disseminated among nearby vehicles for safety and traffic efficiency purposes through distributed consensus.

3.1 Introduction

While a connected vehicle can be informed if an emergency vehicle is approaching far away or a smart intersection can consider the estimated arrival times of connected vehicles to schedule those vehicles and increase the intersection throughput, information security becomes a paramount concern. Such vehicle inevitably make control decisions based on information from other vehicles and external sources. In this scenario, the vehicular network is especially vulnerable to false information attacks. Fake messages can create problems like longer time required to reach destination, more gas consumption than needed, traffic jams, and even collisions. For instance, taking advantage of the connectivity, an attacker may compromise vehicles to lie about its own vehicle's position and speed, to make false warnings about a non-existent accident thus leaving an empty street for itself to freely pass through, or to report fake arriving time to a traffic signal control system causing congestion/traffic jams or even to ease criminals' getaways from crime scenes [100].

To date, research in automotive security has addressed many different perspectives. The goals include secure communication protocols integrated with existing standards and protocols at the external network layer, Intrusion Detection Systems (IDSs) and firewalls at the gateway layer, lightweight message authentication and encryption at the in-vehicular network layer, and Hardware Security Modules (HSMs), secure boot, and secret key control at the component layer.

Nevertheless, even with good security mechanisms at each of those layers, it is necessary to address security at the application layer, especially for application information. In this section, we address this issue through Byzantine-tolerant distributed consensus on application information. Several example use cases include:

Dynamic maps: weather condition, parking availability, EV charging station availability, etc. are all important information comprising a powerful dynamic map. Information reported by a single device/vehicle is not complete and reliable.

Traffic alerts: to provide trust in traffic condition warnings such as collision, congestion, and emergency vehicle(s) approaching, and to avoid inappropriate reactions, distributed consensus can be used to provide trusted alert dissemination.

Intersection management: the next-generation transportation system such as Intelligent Traffic Signal System (I-SIG) relies on vehicles' reported speed and location information to estimate the queuing line, and assigns green/red light time as needed. However, attacks [101] have been found to potentially cause serious problems.

Despite the conceptual appeal of this approach, realizing distributed consensus poses many challenges in connected vehicle systems. One challenge is *safety-critical operation* in the face of *real-time constraint*. For example, if an emergency vehicle needs to take priority at an intersection, this information is only important when it is approaching the intersection. After the vehicle passes, the information is no longer useful. However, forcing consensus within a short period of time could lead to incorrect decisions that may make the situation worse than without connected vehicle system. The second challenge is the

high mobility and *communication loss/delay*, which could significantly affect the ability to reach consensus among devices. The third major challenge is the *lack of trust* in connected vehicle systems. Compromised vehicles with valid credentials will appear as trusted entities [102], which is a difficult situation to handle. Moreover, it is not necessary for compromised vehicles to always behave incorrectly. They may behave as healthy devices at one time and act incorrectly at another time, thus making it hard to rely on reputational trust. To address these challenges, we introduce a distributed consensus algorithm based on *Proof-of-Eligibility (PoE)*. To the best of our knowledge, this is the first work addressing distributed consensus in the face of the challenges listed above.

3.2 A Motivating Example

In this section, we use dissemination of an accident alert as an example to illustrate how our proposed PoE-based consensus algorithm tackles the information security problem in connected vehicle systems. Figure 3.1 demonstrates a fake warning reported by compromised vehicle K. Each vehicle is labeled with letters as its name. Vehicles in green and yellow, representing different brands of vehicles, are honest nodes following the protocol and vehicles in red representing compromised nodes are trying to attack. Without a cooperative evaluation approach, a vehicle can only rely on its received data and local plausibility check as discussed in [35, 103, 104, 105] to make local decisions. Vehicles that are not able to "see" the crossing, may believe the false alert and could potentially reroute and transmit false information to further vehicles, if K is a compromised vehicle with valid credentials.

Our algorithm approaches this problem by using the concept of event reports, whose content could be an unconditional lane shift, slowing speed/congestion, observation of emergency vehicles, crashed vehicles, abnormal behaviors of neighboring vehicles, etc. In order for a created event report to be accepted by other vehicles in the network, a consensus group is formed with a group of eligible vehicles, who can solve a Proof-of-Eligibility (PoE) puzzle, to cooperatively evaluate the report content and reach consensus on whether

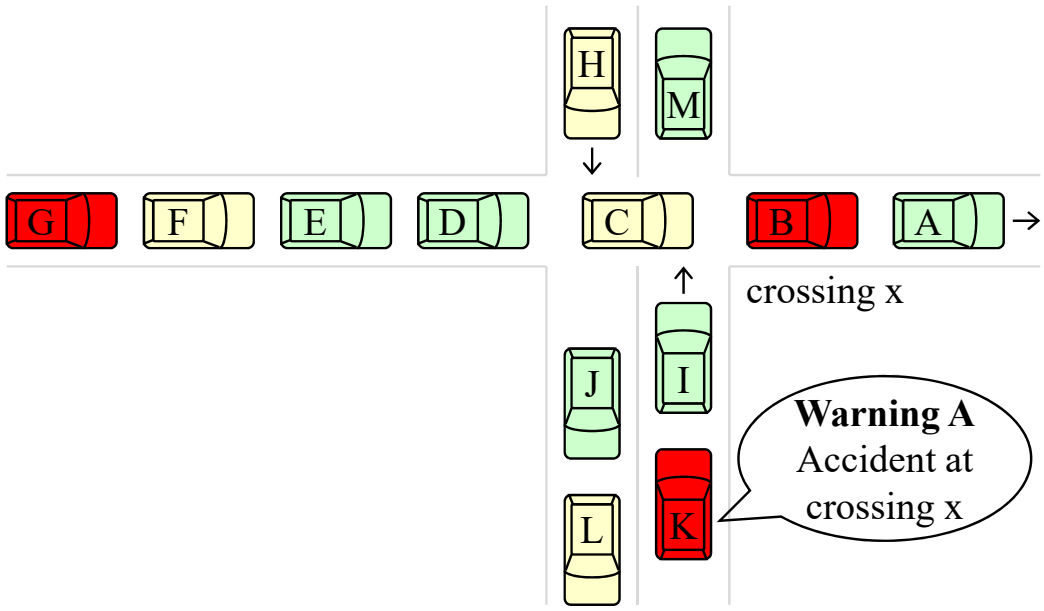


Figure 3.1: Example Scenario of a Fake Report

or not the report is true. Only after that, the true report will be broadcast with group members' signatures. Upon receiving the signed report, other vehicles will react accordingly after verifying the attached signatures.

As depicted in Figure 3.2, after vehicle K broadcasts an event report about an accident at crossing x, all vehicles within communication range (A-L) are able to receive the report. They try to solve a PoE puzzle attached in the event report to obtain a shared secret key. PoE is a set of consistency checks, which aims to prove that a vehicle is authentically relevant and eligible to participate in the cooperative evaluation of a reported event. The PoE puzzle is based on the local environment and can, therefore, only be solved by vehicles that are within close range of the event. PoE lessens the difficulties and shortens the delay of distributing shared keys among a temporarily formed group of moving vehicles. Let us consider a worst case scenario that both compromised vehicles B and G are able to solve the PoE puzzle and join the consensus group. During the consensus stage, compromised vehicles send false opinions agreeing with the fake report that there is an accident at crossing x while the honest vehicles dispute the report. Compromised vehicles B and G may

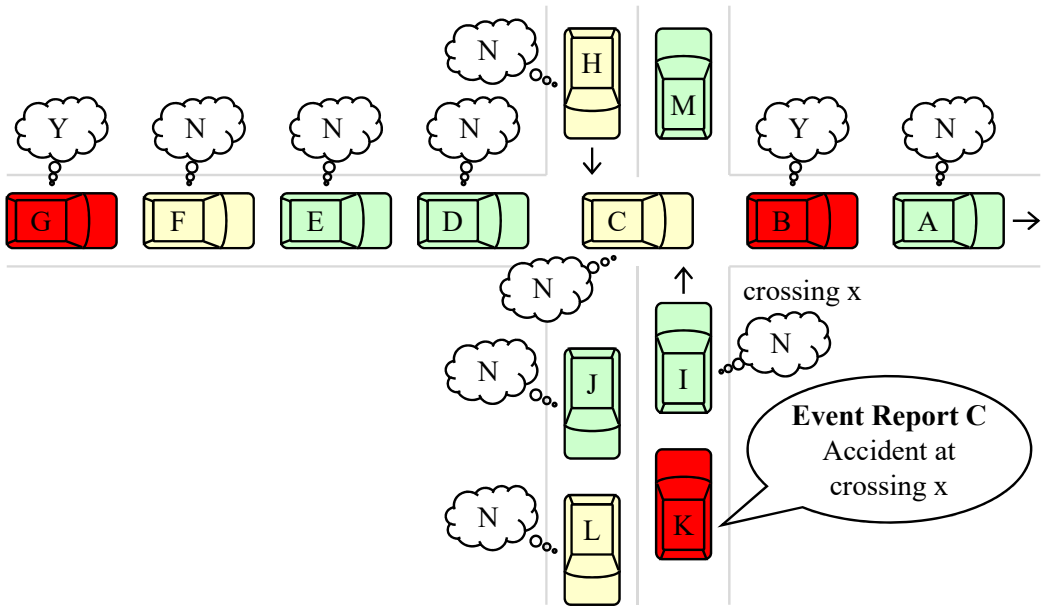


Figure 3.2: Example Scenario of a Fake Report with BFCV (where vehicles B, G, and K are compromised)

also drop the consensus message received from other members to affect the group evaluation. However, as we will see later, with a minority of compromised vehicles participating, false consensus can never be reached with the BFCV Algorithm. In most cases, the group of honest vehicles will reach agreement that the report is false and disseminate a signed message repudiating the report.

In the above, the BFCV Algorithm description has been simplified to briefly introduce the general concept. A detailed algorithm description is provided in Section 3.4.

3.3 Problem Formulation

3.3.1 Assumptions

We assume active (engine-on) vehicles communicating with each other wirelessly. Vehicles routinely exchange information and monitor the environment, following these four steps: *Detection* – a vehicle detects new events (traffic condition, abnormal behavior, etc.) by receiving data from on-board sensors and surrounding vehicles. *Dissemination* – if a detected event is critical, a vehicle creates and broadcasts an event report to other nearby

vehicles. *Decision* – upon receiving an event report, a vehicle evaluates the content of the report and makes a decision to accept it or not. *Reaction* – if an event report is accepted, a vehicle takes the corresponding action(s) such as to brake, accelerate, switch lanes, change routes, disseminate the event report, etc.

We mainly focus on dissemination and decision stages, which are the cornerstones of achieving reliable final reactions. Our goal is to identify potential security violations when attackers have the ability to tamper with information in messages and to mitigate the impact in a timely way. Before describing the threat and system models, we first state some assumptions: (1) The distributed system is asynchronous (unbounded communication delays) and we ensure the safety of our consensus protocol. However, liveness is not guaranteed unless enough messages are received within a time upper bound. (2) An attacker may exhibit compromised behavior at any point in time and remain benign at another time i.e., any vehicle in the network at time period $[t_i, t_{i+1}]$ can be compromised, even if it is behaving normally at time period $[0, t_i]$. (3) We assume that adversaries have limited computing power so that they cannot break the encryption and digital signatures. In other words, the cryptographic algorithms adopted are computationally secure. (4) Vehicles have public key certificates signed by trusted entities such as NHTSA [106] and/or vehicle manufacturers. (5) Private keys cannot be obtained by an attacker without a physical attack. However, by compromising a vehicle through software, an attacker can use an API to sign fake messages but does not know the actual key. This prevents remote attackers from stealing a valid private key from one vehicle and using it within a different vehicle or device.

3.3.2 System Model

We consider a set of vehicles that communicate by sending messages. We assume an unreliable communication medium where messages can be lost or delayed. A vehicle cannot receive other vehicles' messages if they are outside of the communication range (e.g., 200–300 meters for DSRC). Each vehicle can identify the sender of every message it receives

by the sender's unique public key.

We assume that consensus begins with a vehicle generating an event report about conditions it observes on the road. The challenge, as expressed earlier, is for a set of vehicles nearby the event, that were previously unknown to each other, to form a group and reach consensus on whether the event report is accurate in a timely fashion despite the presence of compromised vehicles in the event area. Each vehicle that is nearby the reported event can form an opinion about whether the event report is accurate. We assume that non-compromised vehicles can correctly determine the accuracy of a report most of the time but occasionally a non-compromised vehicle might produce a wrong evaluation due to inaccurate or ambiguous sensing. We refer to vehicles that are not compromised but produce a wrong evaluation of an event report as *incorrect*. In this situation, it is useful for vehicles to learn a group opinion of the report accuracy to verify that their local sensor values are correct.

3.3.3 Threat and Fault Models

We are primarily concerned with attackers who compromise vehicles with valid credentials, and exploit improper/incomplete authorization checks. We adopt a very general threat model, where a compromised vehicle behaves arbitrarily (known as the Byzantine fault model), i.e. it may arbitrarily deviate from the protocol execution and can influence the data sent to communication channel. Through compromised software running on a vehicle, an attacker can broadcast any random or customized false data to the network, but cannot modify others' signed messages or otherwise interfere with others' message creation. In the most basic form, after successfully compromising a vehicle, typical exploits include withholding messages and sending out false data or irregular messages to others. Such attacks are successful when attackers can obtain compromised vehicles' valid certificates and credentials. Otherwise, the sent information cannot pass the authentication checks.

We assume that the number of compromised or incorrect vehicles within a small area is

limited. To be specific, we assume that $f < u_{\min}/3$, where f is the number of vehicles in an area that are compromised or incorrect and $u_{\min}$ is a configurable parameter. Later, we will discuss how PoE puzzles can be used to limit participation in the consensus procedure to only those vehicles that are within the vicinity of the reported event area. This helps to limit the number of compromised vehicles that can influence the consensus, allowing f and $u_{\min}$ to remain fairly small. This, in turn, improves the overall efficiency of the consensus operation and allows consensus to be completed faster, as compared to larger consensus groups that would be required for higher values of f. This also allows the total number of compromised vehicles in the network, beyond the vicinity of the reported event area, to be much larger than f.

Sybil attacks, first introduced in [107], are also a critical problem. An attacker launches a Sybil attack by creating multiple non-existent vehicles with valid identities spreading false information in the network. Various Sybil detection methods have been studied in the past, for example based on: directional antennas [108], received signal strength indicator (RSSI), fingerprinting [109], and interference-aware RSSI-based localization [110]. We assume that Sybil attacks are prevented by existing methods and so we do not consider them herein.

We also assume that there is no large-scale prearranged collusion between compromised vehicles to share answers to PoE puzzles. So, for example, a compromised vehicle does not predetermine PoE puzzles and distribute the answers to large numbers of other compromised vehicles. However, within a consensus group, compromised vehicles can collude arbitrarily (the Byzantine fault model, including collusion, applies within a consensus group).

3.3.4 Consensus Properties

Let Π represent a set of vehicles running a consensus algorithm on some event report R and let v_i be some vehicle in Π. Let x denote the correct evaluation result of R and $\bar{x}$ denote

the opposite (incorrect) evaluation result. Finally, let $S_{ix} = \{v_j : v_i \text{ has heard } x \text{ from } v_j\}$, let $S_{i\bar{x}} = \{v_j : v_i \text{ has heard } \bar{x} \text{ from } v_j\}$, and let $S_i = S_{ix} \cup S_{i\bar{x}}$. S_{ix} contains the vehicles from which v_i has an evaluation result of x, $S_{i\bar{x}}$ contains the vehicles from which v_i has an evaluation result of $\bar{x}$, and S_i contains the vehicles that v_i knows about.

1. *False Consensus* occurs when the following condition holds:

   FC: $\exists\, Q \subseteq \Pi$ such that $|Q| > \frac{2u_{\min}}{3}$ and $\forall v_i \in Q$, $|S_{i\bar{x}}| > \frac{2|S_i|}{3}$ and $|S_i| \geq u_{\min}$ and $\forall v_i, v_j \in Q, i \neq j$, $S_i = S_j$.

   Condition FC occurs when there is a group of vehicles of size greater than $\frac{2u_{\min}}{3}$ that all have heard the wrong evaluation result from more than 2/3 of the vehicles they have heard from and that also agree on the group membership at the end of algorithm execution. If this situation occurs with our BFCV algorithm presented later, a false event report will be disseminated. It is important to prevent this outcome.

2. *Correct Consensus* occurs when Condition FC does not hold *and* the following condition holds:

   CC: $\exists\, Q \subseteq \Pi$ such that $|Q| > \frac{2u_{\min}}{3}$ and $\forall v_i \in Q$, $|S_{ix}| > \frac{2|S_i|}{3}$ and $|S_i| \geq u_{\min}$ and $\forall v_i, v_j \in Q, i \neq j$, $S_i = S_j$.

   Condition CC occurs when Condition FC does not occur and there is a group of vehicles of size greater than $\frac{2u_{\min}}{3}$ that all have heard the wrong evaluation result from more than 2/3 of the vehicles they have heard from and that also agree on the group membership at the end of algorithm execution. Note that if there are two large enough groups formed where one group agrees on the incorrect result and the other group agrees on the correct result, we still consider this to be false consensus. So, correct consensus is a large enough group agreeing on the correct result and the

membership while no other large enough group agrees on the incorrect result. This is the ideal outcome for a protocol.

3. *No Consensus* occurs when $Q \subseteq \Pi$, satisfying Condition CC or Condition FC.

   This describes the situation where there is no consensus reached on either the evaluation result or the group membership or both at the end of algorithm execution. This situation would apply to algorithms that need a result within a certain time bound and terminate algorithm execution if it takes too long without reaching consensus. No consensus is preferable to false consensus but is still an outcome we would like to minimize. If the rate of no consensus is too high, valid event reports will not reach vehicles in a timely way, and this could potentially cause systemic problems.

3.4 BFCV Algorithm

3.4.1 Design Overview

BFCV has three features that differ from existing consensus algorithms that are targeted at connected vehicle systems.

First, it provides fast, reliable consensus group formation and shared key distribution without privileged members. The algorithm does not require trusted set up or leader election and only relies on very basic cryptographic assumptions. Each vehicle running on streets is considered as an untrusted entity equipped with valid credentials and some shares of knowledge describing the environment (traffic, weather, pedestrian, road-signs, etc.). Based on the assumption that at run time, the system does not know which entity is trustworthy and which is not, we do not follow the leader-election paradigm to construct evaluation groups. Instead, we use a set of challenge problems to perform plausibility checks, only allowing entities with sufficient proof of related knowledge and presence nearby the event location to join the cooperative evaluation.

Second, in most cases, BFCV guarantees all participating healthy vehicles reach agree-

ment on the information being disseminated. In cases where the number of healthy vehicles is small or there is a very high rate of message loss, no consensus will be reached but false consensus will not occur (see Section 3.4.6 for a proof sketch).

Third, BFCV is agnostic to wireless communication technology as long as it supports inter-vehicle communication and underlying applications. No additional functionality such as remote cloud for computing, secure channel for message exchange, or trusted entities is required – it is fully distributed and self-maintained.

We next present the BFCV algorithm, which is divided into four phases: Report Generation, Proof-of-Eligibility, Evaluation Group Consensus and Report Verification. Notations used in the following sections are defined in Table 3.1.

3.4.2 Event Report Generation

An event report denoted as R_E, where

$$R_E = (RID, E, EType, Cert_i, Q, \mathcal{H}(A), t_R, T_q) \tag{3.1}$$

is generated and broadcast when a vehicle detects an unreported new event, where RID is the report ID, E is the event content, including location and estimated event life time based on criticalness, $EType$ is the event type, $Cert_i$ is the vehicle's certificate, Q is the PoE challenge problem, $\mathcal{H}(A)$ is the hash of computed answers to Q generated by the reporting vehicle, t_R is the event report time, and T_q is the time bound allowed for solving puzzle Q.

In real life scenarios, it is rare that within 200-300 meters, multiple critical events of the same type (such as rear-end collision, vehicle roll-over, etc.) exist. To improve the system efficiency and discourage compromised vehicles from flooding the network with fake reports, we allow only one event report for one $EType$ of event to be created and broadcast at a time. Once a report is broadcast, the reporter can neither join a different consensus group of the same event type, nor create another event report of the same type until the

Table 3.1: Notation Table

v_i	Vehicle i
K_i^+, K_i^-	Public and private key of v_i
$Cert_i$	Certificate of v_i
D_i	Perception data of v_i
S_i	Hash table that records consensus status of v_i for different types of event reports
P_i	Event look-up table of v_i
F_i	Set of PoE challenge problem functions of v_i
E	An event
EID	An event's ID
$EType$	An event's type (collision, congestion, etc.)
R	An event report
RID	An event report's ID
Q	PoE challenge problems
A	Event Reporters' Answers to Q
$\mathcal{H}(A)$	Hash of A
t_R	Time stamp of report creation time
X	Signature
T_c	Time bound for reaching consensus
T_q	Time bound for solving Q, $T_q < T_c$
u_{min}	Minimum consensus group size
u_i	Membership list of v_i

time bound of the consensus protocol for its current report is reached. If a vehicle does not follow the protocol and broadcasts a new event report before finishing the consensus time period, the inconsistency can be easily caught by other honest vehicles when they are solving the PoE challenge.

3.4.3 The Concept of Proof-of-Eligibility

Before an event report can be accepted by other vehicles, it needs a valid group of vehicles to approve it. How vehicles are selected to form a valid evaluation group is the key to our proposed algorithm. We introduce the the concept of Proof-of-Eligibility to address this.

PoE challenge is powerful, but very application specific. In general, there is a large challenge problem pool pre-installed in vehicles. Let Φ denote the pool stored in vehicles, Q denote a selected challenge problem set from Φ, and function f denote a single problem in Q. Every time an event report is created, a Q will be automatically selected from Φ based on the event type, the event reporter's sensor data, the event time, and a randomly generated nonce. Each set Q must consist of problems of the following types:

View - this type of problem proves the vehicle's proximity to the target position and whether it has a potential view of the event. For instance, even if two vehicles are geographically close in distance, the two vehicles might be on two sides of a big building. Then if an accident happens on one side A, the vehicle on the other side is very unlikely to detect it. Thus, we consider that the vehicle on the other side has a close enough position but does not have a qualified view. Problems in this category use features such as position, speed, acceleration, speed limit, moving direction, colors of a nearby building, number of stop signs, etc.

Knowledge - this type of problem proves whether the vehicle has a certain amount of knowledge about the event of interest. For example, if the vehicle itself is at street S_a and the event is about whether the green vehicle moving on street S_a is compromised. If the vehicle does not even observe a green vehicle on S_a, it definitely has no knowledge of the

event. Problems in this category include true or false questions such as whether the vehicle received a BSM (basic safety message) from location A or whether the vehicle's current speed is below 60 mph.

Consistency - this proves the consistency of a vehicle. Even if a vehicle gets all problems from category 1 and category 2 correct, its answers could have been lucky guesses. This type of problem aims to ask a sequence of true or false questions to further check whether the answers have inconsistencies. For example, questions in category 1 may ask the color of a building and a moving direction. Then question in this category may ask whether the vehicle can see another building of another color. However, the vehicle cannot see this color unless it moves in the opposite direction. If the vehicle's answer is true, then it fails the consistency test.

Algorithm 1: Proof-of-Eligibility Challenge

vehicle v_j receives $R_E = (RID, E, Cert_i, Q, \mathcal{H}(A), t_R, X)$ from v_i;
while (v_j *is operating*) $\wedge$ ($s_j[EType] =$ *idle*) **do**
  if $(t - t_R) < T_q$ *and* X *is correct* **then**
    $A' = RID$;
    for $f \in Q$ **do** $A' = concatenateBits(A', f(D_j))$;
    if $(\mathcal{H}(A') = \mathcal{H}(A)) \wedge (t < T_q)$ **then**
      obtain shared key $K_R^- = KeyGen(A')$;
      set $S_j[EType]$ to busy until $(t - t_R) > T_c$ or consensus is reached;
  else drop R_E;

In the proposed algorithm, once an event report is received by a vehicle, it tries to solve the puzzle Q within time bound T_q; in the end, qualified vehicles are able to obtain a seed to feed into their local key generation function thus obtaining a shared secret key K_R^-. Vehicles then use the obtained K_R^- to initiate a $Hello$ message to other group members. By using the PoE puzzle, evaluation groups are able to form at run time without a selected leader, which saves the time spent on leader election and avoids the risk of granting privi-

leges to a compromised leader. In our proposed model, all group members have the same privilege. The procedure of proof-of-eligibility is presented in Algorithm 9.

In practice, not all of the above problem types can be easily implemented, due to limitations of current vehicles. In our prototype, we only implemented category 1 and 2 problems, excluding problems that require a camera and image processing. In our prototype, only problems that can be answered from existing sensors such as speed, acceleration, GPS location, etc., are implemented. Better PoE challenge design is the subject of future research. Emerging technologies will likely help with this. For example some newly emerged light flashing techniques [111] produce light not visible to drivers but allow vehicles equipped with cameras to capture it, and these techniques will work very well for PoE applications.

3.4.4 Evaluation Group Consensus

Reaching consensus in dynamic vehicular networks in a timely fashion is the key feature of our proposed algorithm. As described in the last subsection, a vehicle that successfully solves the PoE challenge broadcasts an encrypted hello message, M_H, with below format, to establish connections with other members:

$$M_H = Enc(K_R^-, Cert_i, R_E, x_i, Sign(K_i^-, R_E, x_i), \tag{3.2}$$

where x_i is v_i's local opinion of the event report value.

Once some connections among group members are established, encrypted consensus messages, M_C, with below format are sent to initiate voting consensus among members:

$$\begin{aligned} M_C = Enc(K_R^-, Cert_i, R_E, u_i, x_i, \\ Sign(K_i^-, R_E, x_i), Sign(K_i^-, u_i), \end{aligned} \tag{3.3}$$

where u_i is the known-member list by v_i. Each member in u_i is represented by its

public key. The signature of $Sign(K_i^-, x_i)$ denotes attesting of the opinion by the sender, and $Sign(K_i^-, u_i)$ denotes attesting of the sender's recognized group member list.

Traditional consensus algorithms require fixed and known membership. However this is extremely hard to obtain in a highly dynamic vehicular network. We perform consensus on the group membership list G and the opinion list O simultaneously, instead of first agreeing on group membership and then initiating opinion consensus among agreed-upon members. Each element in G and O is uniquely linked to a group member which had its hello message received. For example, if a vehicle v_k receives a hello message from vehicle v_j, v_j is added to v_k's membership list, then $G_k[j]$ is initialized to 0 indicating that v_j is now a known member of v_k. O_k gets updated with $O_k[j] = 1$ if v_j agrees with v_k, otherwise, $O_k[j] = 0$. Moreover, $G_k[j]$ is set to 1 if the membership list of v_j is the same as that of v_k. This is realized by comparing the membership list obtained from M_H with v_k's local membership list. A consensus is reached when more than 2/3 of the vehicles in one vehicle's membership list agree both on the membership and the report value (opinion). The more than 2/3 requirement satisfies the well-known bound for Byzantine agreement. At this point, if the group membership size is at least as large as the minimum group size, then a decision message is broadcast to the network.

A time bound for consensus T_c is set to ensure the effectiveness of the algorithm in vehicular networks. If consensus is reached before T_c, the consensus process terminates with a decision message being broadcast. If consensus is not reached before T_c, the consensus process terminates and the related event report is dropped.

Additionally, we introduce a time variable t_{step} such that for every t_{step}, the vehicle broadcasts a message M_C even when there are no consensus messages or hello messages received. This is important when the vehicle's previously sent messages suffer from packet loss and the vehicle becomes disconnected from the other members. The detailed procedure is presented in Algorithm 29.

Finally, in order to tolerate packet loss, delay, and Byzantine behavior, we allow a

vehicle to add another vehicle to its group list when it observes the vehicle in enough other vehicles' group lists even if it did not receive a hello message from the vehicle. The threshold we set for this is more than $\frac{1}{3}$ of the minimum group size to ensure that at least one healthy vehicle has heard a hello message from the new vehicle. In order to keep the code description fairly simple, we do not show this aspect in the pseudocode.

Algorithm 2: Evaluation Group Consensus

v_k obtained K_R^- by solving PoE challenges;
v_k receives a message M from v_j ($j \neq k$), set $t' = t$;
while $(t - t_R) < T_c \wedge$ *(consensus is not reached)* **do**
    if $t \geq (t' + t_{\text{step}})$ **then** set $t' = t$, create and broadcast M_C;
    if M *can be decoded using* K_R^- *and verified* **then**
        if M *is a hello message* $\wedge\, v_j \notin u_k \wedge\, !\Pi_{sync}$ **then**
            add v_j to u_k;
            if $x_j \neq x_k$, then $O_k[j] = 0$, otherwise $O_k[j] = 1$;
            $t' = t$, create and broadcast M_C;
        else if M *is a consensus message* $\wedge\, v_j \in u_k$ **then**
            if $!\,\Pi_{sync}$ **then**
                if $v_k \notin u_j$ **then** send hello message M_H;
                if $x_j \neq x_k$ **then** $O_k[j] = 0$;
                else $O_k[j] = 1$;
                if $u_j \neq u_k$ **then** $G_k[j] = 0$;
                else $G_k[j] = 1$;
                if $|\{l \mid G_k[l] = 1, O_k[l] = 1, l \in u_k\}| > \frac{2|u_k|}{3}$
                $\wedge\, |u_k| \geq u_{\min} \wedge (t - t_R) > T_q$ **then**
                    $\Pi_{sync} =$ true and set consensus flag to true;
                    create and broadcast decision message;
                else $\Pi_{sync} =$ false;

3.4.5 Event Report Verification

A decision message is created if more than 2/3 of the vehicles agree on the same value and on the group membership. A decision message is denoted by:

$$M_D = (R_E, \alpha, CERTs, SIGs), \tag{3.4}$$

where α is the decision result, $CERTs$ is a set of certificates of the group members, and $SIGs$ is a set of signed opinions of the members in u such that for each v_i, $Sig_i = Sign(K_i, x_i)$.

When a vehicle receives a decision message and either it is not part of the consensus group or it is part of the consensus group but has not yet reached a decision, it examines the attached signatures. If the group size is at least $u_{\min}$, all signatures are valid, and more than 2/3 of the signed values agree with the decision value, the vehicle accepts the decision. In this way, vehicles that are not compromised but have the incorrect value will accept the group decision about the event's status. Any message with one or more invalid signatures or a group size less than $u_{\min}$ will be discarded. If multiple different decision messages regarding the same event report are received by a vehicle, it accepts the valid decision message with the longest signature chain and rejects the others.

3.4.6 Proof Sketch of Protocol Correctness

As is typical for distributed consensus protocols in challenging environments, the PoE protocol guarantees safety but not liveness. However, liveness is demonstrated through our simulation experiments described in Section 3.5.

Our main safety property is that false consensus does not occur as long as less than 1/3 of the vehicles in the vicinity of the event are compromised or incorrect. Thus, the only possible outcomes of the protocol are correct consensus and no consensus. This is detailed in the following claim and proof sketch.

Claim: Let the minimum consensus group size be $u_{\min}$. As long as the number of compromised vehicles and incorrect vehicles in the area of an event report $R_E(t)$ between the time of the report t and the time $t + T_c$ is less than $u_{\min}/3$, then false consensus cannot occur.

Proof Sketch:

The proof of eligibility challenge plays a fundamental role in ensuring safety. Only vehicles that can observe the area of the event report are capable of passing the challenge. This prevents compromised vehicles from *outside* of the event area from participating in the consensus. Thus, only compromised vehicles within the area of the report during the time that the consensus protocol is executed need be considered.

Additionally, we assume that the number of healthy vehicles in the event area that do not correctly verify the status of an event report is very small so that the total of compromised and incorrect vehicles in the vicinity of the report is less than $u_{\min}/3$.

False consensus requires agreement on the wrong value of an event report (e.g. "no accident" when an accident has actually occurred) and agreement on group membership. This could possibly occur in two situations: 1) when a vehicle correctly reports an event but enough compromised and incorrect vehicles within the formed consensus group conclude the event did not occur, or 2) when a compromised or incorrect vehicle falsely reports an event and enough other vehicles support the false report during the consensus procedure.

In either situation, there are two possibilities; either a consensus group of size at least $u_{\min}$ is formed for the event report or no large enough group is formed. If no large enough group is formed before time $t + T_C$, then no healthy node can broadcast a decision (see Lines 29–32 of Alg. 2 pseudocode) and the event report is dropped (this is a no consensus outcome).

If a large enough consensus group is formed, this means there are fewer than $u_{\min}/3$ compromised or incorrect nodes within the group that support the wrong event status. Thus, there are simply not enough nodes to broadcast the false evaluation value for any healthy node to accept it, since that would require more than $2u_{\min}/3$ false evaluations to be broad-

cast by distinct nodes. In this situation. If enough nodes that receive more than $2u_{\min}/3$ correct evaluation results also agree on membership of the consensus group, the result is correct consensus but if there are not enough nodes that agree on the membership, the result is no consensus. However, in neither case, can false consensus occur.

3.5 Evaluation

3.5.1 Implementation

We implemented a prototype of BFCV in C++, which can simulate different scenarios by changing the map and system parameters. It is build on top of Veins [95] which provides a comprehensive suite of models of IEEE 802.11p, IEEE 1609.4 DSRC/WAVE and obstacle shadowing. We add additional layers to simulate packet loss/delay, cypto schemes supporting 128, 192, and 256 bit ECDSA keys for encryption, signing and verification, SHA-256 as the hash function. Our prototype consists of approximately 3500 lines of written code. Experimental maps are obtained from OpenStreetMap (OSM) [112]) with manual corrections of speed limit, traffic lights, number of lanes on the road, etc. to improve the accuracy. Vehicle mobility and routes are computed based on demand definition and shortest path algorithm using SUMO [113].

Different from previous works, the prototype includes realistic aspects of the vehicle dynamics (safe distance, mass, dimensions, vehicle types, braking distance, traffic lights, etc.), detailed modeling of the communication network, and real-life street maps with varying scales.

3.5.2 Simulation Results

Experiment Scenario

Our evaluation is conducted in a simulated midtown area of a major city in the U.S. with a capacity of around 700 moving vehicles (see Figure 3.3). We simulate a worst-case sce-

Figure 3.3: Evaluation Scenario - Urban

nario where compromised vehicles behave honestly when there is no event to be reported. Thus, it is very hard for honest vehicles to catch bad behaviors prior to an event report. In the simulation scenario, a collision happens at a random time, and honest vehicles that detect the event create and broadcast ”collision occurred” event reports leaving others to evaluate them. We also have compromised vehicles broadcast conflicting event reports saying ”collision cleared” at the same time. Thus, there can be several consensus executions happening at the same time among different groups of vehicles to try to reach agreement to accept one of these conflicting reports. We also have compromised vehicles drop, delay or not send messages, and submit wrong opinions for evaluation.

BFCV Evaluation Results

Unless otherwise noted, the following parameters were used in all experiments: $T_q = 5s$, $T_c = 14s$, $u_{\min} = 7$, vehicle density $= 250$ and beacon message frequency $= 10$Hz. Natural packet loss and delay (not including compromised vehicles' behavior) were simulated such that messages were randomly dropped at receiving vehicles with a drop rate of 15% and packets were randomly delayed within a range of 100ms - 1500ms. The communication range among vehicles was set to $300m$ based on NHTSA's proposed rule [114]. We

use the following two metrics to evaluate the latency of BFCV:

- *Consensus Time:* the time spent on reaching consensus on a single event report. This starts from the event report creation time and ends when there is a decision message received by every member of the group contained in the message.
- *Decision Time:* the time spent on ultimately reaching consensus. This starts from the first event report creation time and ends when there is a decision message received by every member of the group contained in the consensus message. Note that this could involve multiple consensus attempts if the first attempt does not produce a consensus.

We first evaluated how BFCV's performance varies with minimum group size and vehicle density. We ran simulations with 10% of vehicles compromised, varied $u_{\min}$ from 4 to 10 with an increment of 1, and varied vehicle density from 50 to 450 vehicles with an increment of 100. 50 simulation runs were done for each parameter combination, where a single consensus period was simulated in each run. The possible results of each run are: Correct Consensus (CC), False Consensus (FC) and No Consensus (NC). Note that, as described above, there can be multiple consensus executions happening concurrently for "collision occurred" and "collision cleared" event reports. In case multiple large enough consensus groups succeed in reaching consensus, we record the result as FC as long as at least one of the groups agreed on "collision cleared". The results are shown in Figure 3.4.

Figure 3.4(a) shows the average consensus time in seconds vs. minimum group size and vehicle density. Note that NC outcomes are not included in the average, because there is no definite termination of the consensus in those cases. Not surprisingly, consensus time increases with both minimum group size and vehicle density since an increase in either parameter will cause the number of messages exchanged by the algorithm to increase. Figure 3.4(b)(c)(d) shows the different consensus outcomes vs. the two parameters. Note that, if the minimum group size is too small, compromised vehicles can form a group and reach false consensus. Also, if the vehicle density is too low, there are not enough vehicles

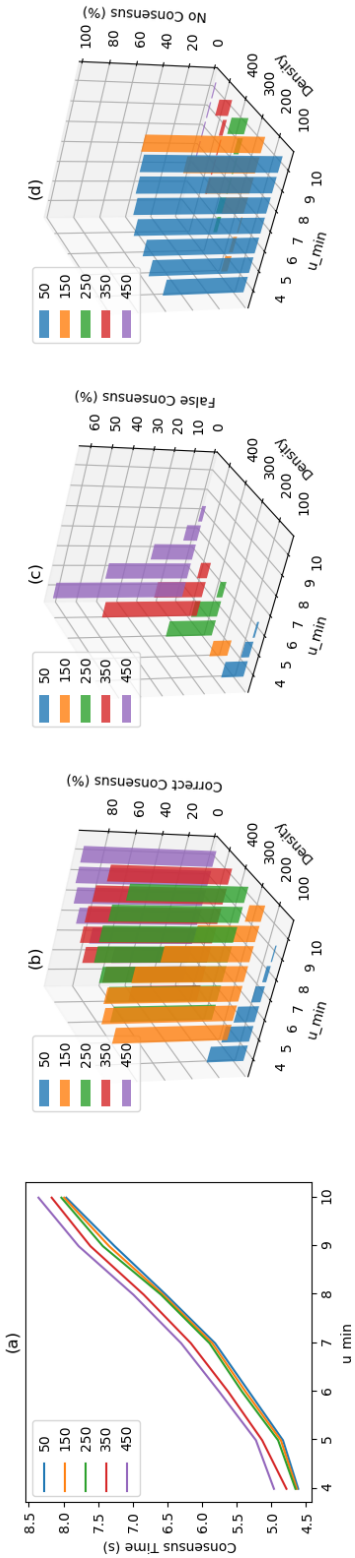


Figure 3.4: Consensus Time and Consensus Result vs. Minimum Group Size and Vehicle Density

Table 3.2: Performance vs % of Compromised Vehicles

Mal_V	CC	FC	NC	AvgConsensusTime
5%	100%	0%	0%	5.228s
10%	99%	0%	1%	5.701s
15%	99%	0%	1%	6.206s
20%	99%	0%	1%	6.718s
25%	97%	0%	3%	8.542s
30%	93%	0%	7%	9.325s
35%	88%	0%	12%	10.446s
40%	79%	0%	21%	13.118s

in the event area to form a consensus group, and this leads to a high rate of NC outcomes. However, for a fairly wide range of group sizes and vehicle densities, there are zero FC results and a very low rate of NC outcomes. These results demonstrate that the choice of $u_{\min}$ should be based on both adversarial assumptions and expected vehicle density.

We also evaluated BFCV's performance versus the percentage of compromised vehicles with $u_{\min} = 7$ and a vehicle density of 250. For percentages from 5% to 40% with 5% increments, we repeated the simulation 50 times. The results are shown in Table 3.2. From the table, we can see that as the percentage of compromised vehicles increases, the percentage of CC decreases from 100% to 79%. However, even with 40% of the vehicles in the network being compromised, the BFCV Algorithm did not experience a single false consensus outcome.

We also evaluated how well BFCV handles failure to reach consensus (NC outcomes). Instead of stopping the simulation immediately after the single-round consensus timeout, T_c, occurred, we extended the simulation if consensus was not reached the first time. If a report evaluation fails to reach consensus within time T_c, then our algorithm drops the report. However, if this occurs in the simulation, another honest vehicle nearby will submit a new event report for consensus. By extending the simulation time, we examined whether the BFCV algorithm can recover from NC outcomes. In this case, we recorded the final de-

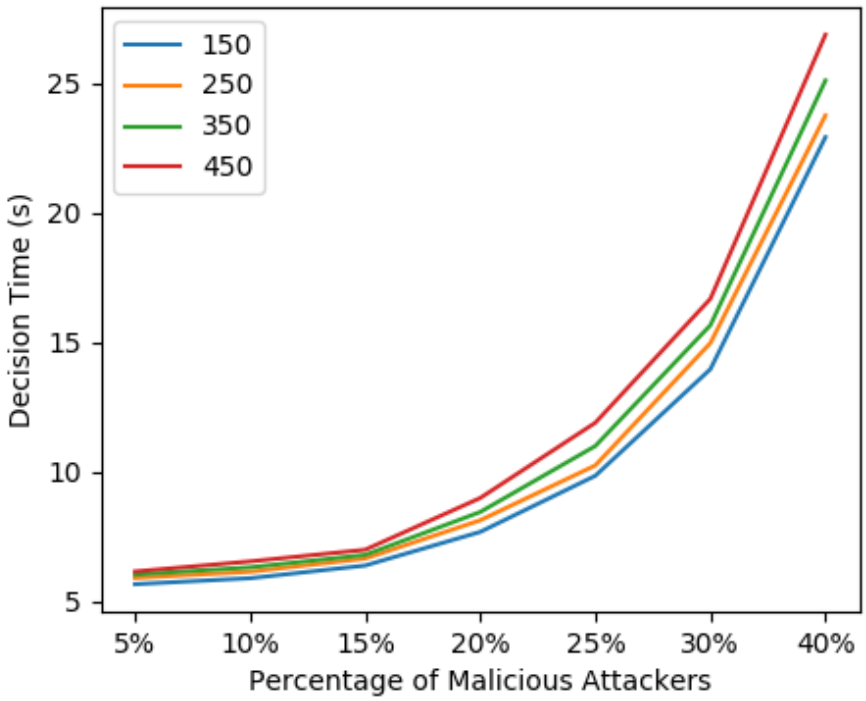


Figure 3.5: Decision Time vs % Compromised Vehicles

cisions, i.e. whether there was ultimately a correct decision made after an event happened, possibly after more than one consensus attempt.

Figure 3.5 shows the average decision time for different compromised vehicle percentages and different vehicle densities with $u_{\min} = 7$. When the percentage of compromised vehicles was low, the decision was made very quickly with an average that is well below the single-round consensus time bound T_c. However, as the percentage of compromised vehicles was increased, the time spent on evaluation rose. Note that, in some cases, the average decision time was close to or exceeded $T_c = 14s$, implying that more than one round of consensus was some times needed for those cases.

Comparison Results

We also simulated two related protocols, DC [40] and PoR [39], and evaluated them under the same experiment conditions. These protocols both use threshold-based voting, which

is the most widely-used prior approach. Two metrics are introduced to compare the results:

- *Percentage of Vehicles Taking Action:* the percentage of vehicles in the network that reach a correct decision about the event report and take action to avoid the accident location
- *Average Commute Time:* the average simulation time that vehicles take to reach their destinations (vehicles not taking action to avoid the accident location experience a longer commute time due to backups around the accident site)

There was no explicit method described in [39] to set the threshold for the PoR algorithm. However, it should be based on the report criticality and the network status, which is similar to the minimum group size in our proposed algorithm. Therefore, we set both of these parameters to 7 in these simulations. The DC algorithm provides an explicit method for dynamically adjusting its threshold value, which we adhered to in our DC implementation. Other parameters of BFCV were the same as in the previous experiments.

Figure 3.6 depicts the percentage of vehicles taking action as the simulations progressed when 5% and 15% of vehicles were compromised, respectively. There are two main reasons why BFCV performed better than DC and PoR. First, with BFCV, vehicles make a group decision and act accordingly. For DC and PoR, each vehicle makes its own decision when enough endorsements from other vehicles are collected and, thus, not all vehicles make the same decision. In particular, since there are both "collision occurred" and "collision cleared" reports being circulated at the same time, some vehicles collect enough votes for "collision cleared" and reach the wrong decision even though most vehicles reach the correct decision. Second, BFCV uses PoE, which prevents compromised vehicles from outside the event area from participating. Since DC and PoR cannot verify location information of vehicles, they cannot filter out fake reports from compromised vehicles anywhere in the network that falsely report their location as being near the event, which increases the chances that enough votes can be collected to accept a fake report.

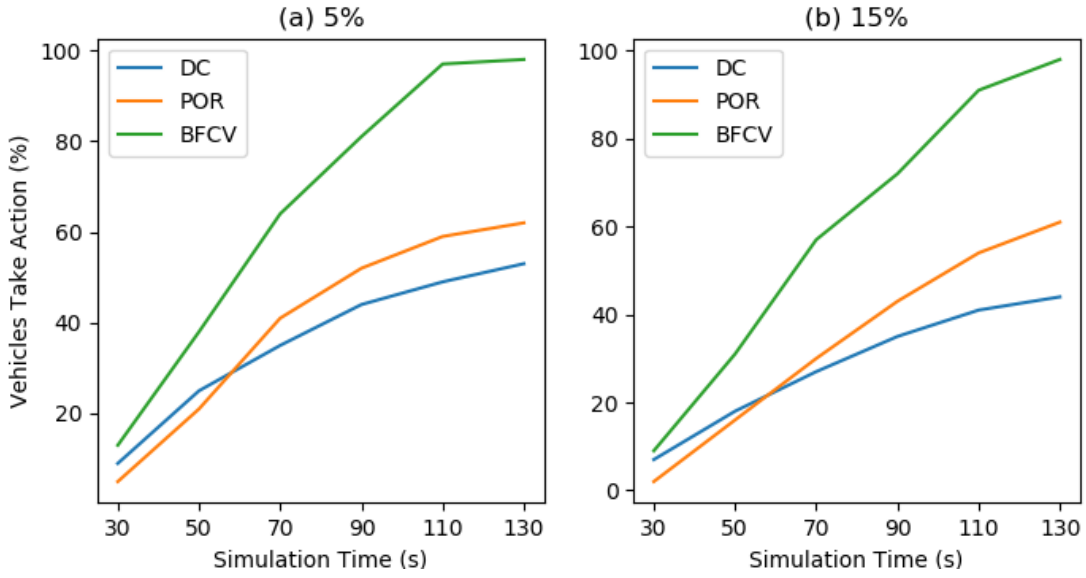


Figure 3.6: Vehicles Taking Action (%) vs Simulation Time

Table 3.3: Avg. Commute Time vs. % of Compromised Vehicles

Mal_V	DC(s)	PoR(s)	BFCV(s)
5%	171	179	155
10%	194	197	162
15%	244	223	183
20%	286	248	203

Table 3.3 depicts average commute time versus percentage of compromised vehicles ranging from 5% to 20%. For vehicles that were stuck in the simulation area at the end of the simulation, for the purposes of computing an average, we assigned them a commute time of 300 sec. From the table, we see that BFCV produced 15-22% lower commute times than PoR and 10-40% lower times than DC. As the percentage of compromised vehicles increased, most of the vehicles actually ended up being stuck for the DC Algorithm (avg. commute time approached 300 sec.), and a significant number were also stuck with PoR, while most of the vehicles actually reached their destinations during the simulated interval with BFCV.

3.6 Chapter Summary

We presented BFCV, a distributed consensus algorithm based on ”proof-of-eligibility”, which achieves Byzantine agreement with unknown group membership and unreliable communication channels and is targeted at vehicular network environments. BFCV leverages the unique characteristics of moving vehicles and cryptographic primitives to prevent a large number of compromised and unrelated vehicles from joining the consensus group. This significantly speeds up the consensus procedure, providing a new paradigm of Byzantine-tolerant fast consensus for connected vehicles.

CHAPTER 4
COOPERATIVE TASK-ORIENTED GROUP FORMATION FOR VEHICULAR NETWORKS

As the methodology proposed in chapter 3 is able to defend a vehicle from making decisions based on unproven information, in this chapter, we move a step further and focus on how to leverage the power of cooperation as well as enriched information sharing for better task completion in the vehicular networks.

4.1 Introduction

With advancements in information and communication technologies in modern vehicles, the amount of data they generate and the computation capability they possess are both growing rapidly. For example, autonomous vehicles (AVs) can generate between 1.4 and 19 terabytes (TB) of data per hour [115]. Such a large amount of data brings not only opportunities but also challenges in various distributed computation tasks requiring cooperation [116], e.g. reinforcement learning based cooperative driving [117], distributed consensus based false information filtering [118], collaborative active learning [119], etc., since if conducted appropriately, the performance of the applications should increase with the number of participants [3]. For instance, a single vehicle may not be able to capture accurate and full perception data from its own sensors due to imprecision and limited view. Aggregating the diverse views from multiple vehicles can improve the results of many computational tasks by creating larger and richer data sets. Moreover, though on-board computing devices are becoming increasingly powerful [120], computing locally with a very large data set often requires enormous computation and memory resources, which hinders these applications on resource-constrained edge devices such as vehicles [115]. As a result, most proposed solutions have assumed that data is sent to a powerful central server

for computation [1].

Nonetheless, the application requirements discussed above pose challenges for the conventional cloud computing paradigm. It is difficult to guarantee the stringent quality/experience requirements due to a large on-board memory requirement, high latency, and limited backhaul bandwidth [121]. To tackle these challenges, recent research has considered mobile edge computing (MEC) [16, 17, 18], vehicular fog computing (VFC) [13, 19, 20] and vehicular cloud computing (VCC) [22], where computation tasks are offloaded to surrounding vehicles with surplus resources, to address certain aspects of the problem.

There are three critical challenges that are not well addressed in this prior work.

1. Whether tasks being assigned to vehicles can be successfully completed is ignored and, instead, focused on internal dependencies between tasks and which tasks should be assigned to which vehicles. Tasks that are not successfully completed serve no useful purpose but waste valuable computational resources.

2. Cooperative task execution was not considered, i.e. they only considered the offloading of a task from one vehicle to another. As discussed earlier, cooperative task execution is required to not only deal with the very large storage and computational resources required to process the large amount of data generated by modern vehicles, but also to break the barrier of limited local views/perceptions by aggregating information from multiple vehicles.

3. Prior works ignored the trade-off between successful task completion and quality of the cooperative computation results. In general, the result quality of such tasks increases with the size of the cooperating group, however, the larger the group becomes, the less stable is the group's connection, which results in a lower probability of task completion.

Thus, in this chapter, we propose a task-oriented group formation method addressing the above challenges. To our best of knowledge, this is the first work addressing com-

putation task oriented group formation in vehicular networks. Details are provided in the following sections.

4.2 System Model Overview

Our system model is based on a bidirectional multi-lane road scenario. We assume basic communication among vehicles is supported, where each vehicle discovers their neighboring vehicles through periodic beacon messages. As is common in vehicular networks' research, e.g. [122, 25], we assume the beacon messages exchanged among vehicles contain the vehicles' basic information, including location, velocity, and moving direction, so that each vehicle can estimate speed difference, relative distance, and traffic condition/density around it as introduced in [123]. We assume that all vehicles use the same physical mode for transmitting or receiving data, and the precise time is known and traceable. In addition to periodic beacon messages, event/application driven messages are also supported. Finally, we assume there are no malicious vehicles, in that all vehicles follow the protocol. However, abnormal behavior such as packet loss or abnormally long packet delay can still occur. Note that we use node and vehicle interchangeably in the rest of this chapter.

4.3 Problem Formulation

The problem that we consider herein is how to construct a task group that is well suited to carrying out a specific task. We call the vehicles that have computation task requests *Task Vehicles* (TVs), whereas the vehicles that can provide their data and surplus computational resources to the TVs are referred to as *Service Vehicles* (SVs). Note that a single vehicle can serve as a TV at one time and as a SV at a different time but cannot serve in both roles simultaneously. Given a group G that is a candidate to perform a cooperative task, there are two important quantities, which are both random variables: 1) *stay time*, denoted by T_G^{stay}, and 2) *task completion time*, denoted by T_G^{task}. T_G^{stay} is the length of time that all members of G remain in communication range of each other, and T_G^{task} is the length of time that the

vehicles of G need to complete the assigned task (assuming that the vehicles stay together for at least that amount of time).

As shown in 4.1(a)-(c), how likely it is that the task will be completed successfully by a given group depends on the means of the T_G^{stay} and T_G^{task} distributions (as well as their shapes). If the mean stay time is much greater than the mean task completion time, e.g. 4.1(a), the task is very likely to be successfully completed. As the two distributions get closer together, e.g. 4.1(b), the probability of successful completion begins to decrease. Obviously, if the mean stay time becomes less than the mean task completion time, then it is very unlikely that the task will be successfully completed, e.g. 4.1(c). We consider a group viable if:

$$P(T_G^{stay} - T_G^{task} \geq 0) \geq 1 - \epsilon \ , \tag{4.1}$$

where $\epsilon \to 0$. While formally calculating this probability requires knowledge of the joint distribution of the two random variables, in practice any dependence between them is quite weak and they can be treated as independent random variables.

The size of a group is an important factor in the stay time and task completion time distributions and it can also impact the quality of the computational result. Larger groups tend to have a larger task completion time variance and higher communication cost, which for cooperative (non-parallel) computations, can also increase the expected task completion time. However, larger groups also tend to break apart more quickly. From these trends, we see that increasing the size of the group tends to drive the distributions from the 4.1(a) case (very small groups) to the 4.1(c) case (very large groups). Finally, we note that for many applications, e.g. the distributed learning example discussed in 4.5, larger task groups will produce higher quality results.[1]

Summarizing the above discussion, we tend to prefer larger task groups in order to im-

[1]For distributed learning, increasing the number of nodes creates a larger overall dataset and higher computational capability.

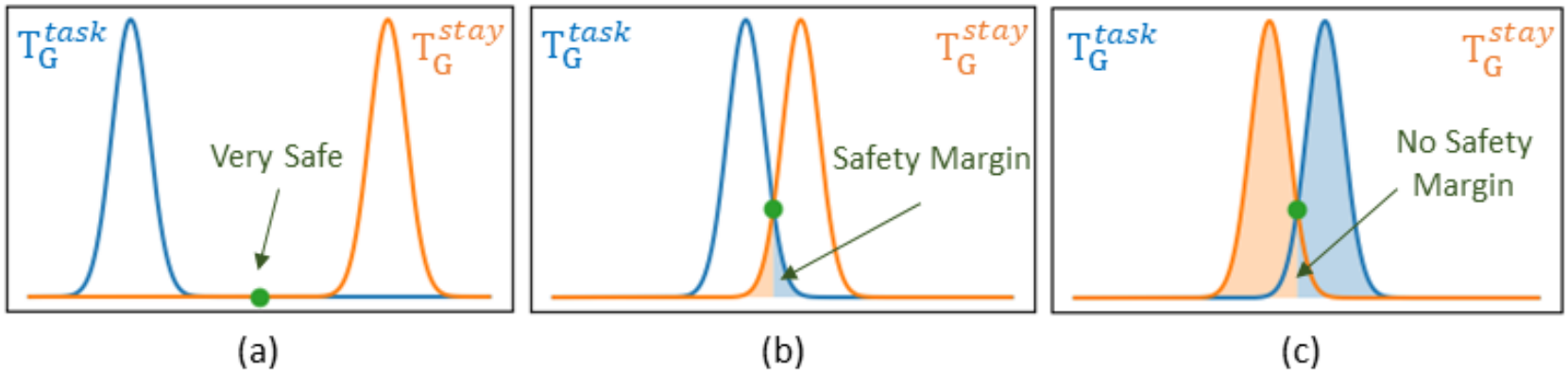


Figure 4.1: Distribution of Stay Time and Task Completion Time. Gaussian distribution used as an illustration – framework does not assume any specific distribution

prove the quality of the computational result. However, larger groups tend to be less likely to complete the task before they break apart. Thus, the goal of our task group formation algorithm, presented in the next section, is: **form as large a group as possible while ensuring that the group is very likely to successfully complete the task**, where "very likely to successfully complete the task" means that the group satisfies Inequality (4.1).

In practice, it is not possible to know the exact probability distributions of the stay time and task completion time of a group. Therefore, in our group formation algorithm presented in the next section, we introduce the notion of a *safety margin*, which is a safe separation between the estimated stay time and estimated task completion time for a group (see Figure 4.1). With an appropriate choice of safety margin, this can be considered as an approximation of the formal group viability condition specified by Inequality (Equation 4.1).

4.4 Task-Oriented Group Formation Algorithm

From the high-level perspective, we solve the problem formulated in the last section by a two-stage task-oriented group formation algorithm (ToG). As for the first stage, neighboring vehicles are clustered together based on relative mobility, connectivity and estimated stay time. The goal is to obtain stable clusters, in which the cluster members can not only stay together longer but also share good connectivity. In this work, we refer to this first stage as the parent clustering stage, where vehicles are clustered together and one parent cluster head (PCH) is selected for each parent cluster. The PCH is responsible for collecting and recording parent cluster members' (PCMs) statuses, position changes, and available resources. When a PCM is assigned a computation task, it becomes a TV and sends a task group formation request to its PCH. This triggers the second-stage – task group formation, where the PCH will select the best suited vehicles to form a task group based on certain criteria.

There are two main benefits of the proposed two-stage approach: a) parent clustering

filters out the neighboring vehicles who have close geo-location but are less likely to stay together, thereby providing good candidates for task group formation; b) allowing a PCH to coordinate different task requests and select the qualified task group members for each of the requests increases efficiency and throughput. The details of the two-stage approach are presented next.

Parent Clustering

It is evident that a stable parent cluster scheme provides a strong basis for efficient task group formation, as the cluster members would stay together longer. Thus, for our parent clustering framework, we adopt the high-level clustering approach from [124], which achieves good cluster stability while maintaining relatively low communication overhead. Specifically, two main techniques are borrowed: 1) capability metrics, which are used to select the cluster head, and 2) substitute heads pre-selection before membership changes, so that both nodes that stay in the original cluster and nodes that are very likely to leave in a short period of time can smoothly settle down with limited effort, which increases the stability of clusters. The capability metrics of [124] that are used to select the cluster head are mainly focused on crossroads; therefore, we add another capability metric, which we refer to as relative distance metric (RDM), which is targeted at broader road types such as highways.

Next, we first define the capability metrics from [124], namely relative velocity metric (RVM) and power loss metric (PLM) and then we define our RDM metric. For vehicle v_i,

$$RVM(i) = \frac{1}{N}\sum_{j=1}^{N} log\left(\frac{v_{max}}{v_{max} - \Delta v_{i,j}}\right) \tag{4.2}$$

and

$$PLM(i) = \frac{1}{N}\sum_{j=1}^{N} log\left(\frac{P^t}{P^r_{i,j}}\right) \tag{4.3}$$

where, v_{max} is an upper bound on velocity, N denotes the number of direct neighbors of v_i,

$\Delta v_{i,j}$ is the velocity difference between vehicles v_i and v_j, P^t is the unified transmission power of all nodes and $P^r_{i,j}$ denotes the received power of v_i from v_j. A smaller value of RVM indicates that the vehicle's velocity is similar to that of its direct neighbors. A smaller value of PLM means that the vehicle is more likely to have shorter communication distance and better channel quality with its direct neighbors.

Vehicles on a highway can have much higher speeds than in crossroads, and the distances between vehicles can be much longer, making the relative distance an important metric. While PLM is also related to distance, it is affected by other factors such as obstacles. Therefore, we add the relative distance metric, defined as:

$$RDM(i) = \frac{1}{N} \sum_{j=1}^{N} log \left(\frac{R}{R - \Delta d_{i,j}} \right) \tag{4.4}$$

where R is the communication range and $\Delta d_{i,j}$ represents the relative distance between node v_i and v_j. A smaller value of RDM indicates that a node is closer to the middle of its neighbors.

The combined capability metric we use is:

$$M(i) = RVM(i) + PLM(i) + RDM(i) \tag{4.5}$$

and the node with the smallest $M(i)$ among the cluster head candidates is selected as cluster head.

As parent cluster stability is the key to forming good task computation groups, we incorporate the neighbor sampling (NS) scheme from [125] to enhance the stability of parent clusters.[2] Additionally, we use a safe leaving distance σ to ensure that a PCH can plan in advance when a node is about to leave. A vehicle whose distance from its PCH is increasing must notify the PCH when it is within a distance σ of moving out of the

[2]Only vehicles within one hop vicinity moving in the same direction as well as a speed difference less than a threshold are considered as neighbor candidates. Refer to Algorithm 1 from [125] for details.

PCH's transmission range. This safe leaving distance guarantees that the PCH can receive a notification about a leave before the connection is lost. Although we added a few other minor enhancements, the parent clustering procedure and maintenance strategy follow the general approach in [124].

Task Group Formation

Our task group formation scheme makes use of estimated stay time and estimated task completion time to choose groups that are well suited for the particular task to be executed. Good stay time and task completion time prediction schemes are the keys to efficient task group formation. We define $T_{i,j}^{stay}$ as the estimated stay time between parent cluster member v_i and v_j such that:

$$T_{i,j}^{stay} = \frac{|\Delta v_{i,j}|(\min\{R, D_{i,head}, D_{j,head}\}) - \Delta v_{i,j}\Delta D_{i,j}}{(\Delta v_{i,j})^2} \tag{4.6}$$

where R denotes the communication range of a vehicle, $D_{i,head}$ ($D_{j,head}$) denotes the distance between v_i (v_j) and the parent cluster head, and $\Delta D_{i,j}$ and $\Delta v_{i,j}$ represent the relative distance and velocity between vehicles v_i and v_j, respectively. In a task group, each vehicle is required to exchange data and computation results with each of the participants in a distributed fashion. Accordingly, a group G can be considered non-functional once the first pair of vehicles loses communication and we therefore estimate the stay time as:

$$\hat{T}_G^{stay} = \min_{i,j \in G}(T_{i,j}^{stay}) \ . \tag{4.7}$$

Different from stay time prediction, task completion time estimation is application specific. Instead of discussing the specific estimation scheme in this section, a distributed learning based application example is provided in Section 4.5.1. We will use the general notation $\hat{T}_G^{task}$ to represent estimated task completion time for a group G in this section. As different vehicles may be equipped with different computing capability, we use κ to represent the

task computation rate of a vehicle. The larger the κ is, the faster a vehicle can compute its task. Let H denote the difference between estimated group stay time and task completion time. We only consider task groups with H larger than the safety margin T_{th}, i.e.

$$H = \hat{T}_G^{stay} - \hat{T}_G^{task} > T_{th} \tag{4.8}$$

Setting $T_{th} > 0$ accounts for the fact that $\hat{T}_G^{stay}$ and $\hat{T}_G^{task}$ are only estimates and we want to ensure that the task can be completed with high probability in its assigned group.

Vehicles in a parent cluster can be in one of three states: available, TV, or SV. PCHs can neither submit any computation request like a TV nor participate in any task computation as a SV. When a PCM has a request for computation assistance, it enters state TV and sends a request to its PCH. As discussed in Section 4.2, a vehicle cannot act as a TV and a SV at the same time. Thus, once a vehicle changes its state from available to TV, it cannot service other tasks' computations until its current request is fulfilled or dropped. A vehicle that is currently serving a request cannot submit a computation request until it is done with its current request, i.e. a vehicle cannot change state directly from SV to TV. Rather, when the vehicle is done servicing a request, it changes its state to available and, only then, can it submit a computation request.

The Algorithm 5 provides the details of the proposed task group formation algorithm. Each PCH maintains a member information table (MI) to keep track of PCMs' mobility and status information, periodically updated by intra-cluster messages. Upon receiving a task request, a PCH will check if there are available PCMs that can service tasks and attempts to form a task group. Initially, PCH starts from the assumption that all available PCMs within communication range R to the requester (TV) and $R - \sigma$ to the PCH are able to service the task, thus adding them to an empty group list G. Then, estimated stay time $\hat{T}_G^{stay}$ and estimated task completion time $\hat{T}_G^{task}$ are computed based on the information in MI. If H, as defined in Equation 4.8, is larger than the safety margin T_{th}, the PCH will

Algorithm 3: Task Group Formation

PCM info table: MI, available PCM $v_i \in$ MI, task vehicle: v^t;
while *PCH receives a task request* $\wedge$ *available PCMs* **do**
    initialize an empty group list $G = [\,]$;
    for *all available* v_i **do**
        if $D_{i,t} < R$ *and* $D_{i,head} < R - \sigma$ **then** add v_i to G;
    if $|G| < C$ **then**
        send task drop notification to v^t;
        go back to the start of WHILE loop;
    else calculate H;
    while $H < T_{th}$ **do**
        $G = G - \underset{v_j \in G}{\operatorname{argmax}} H$, recalculate H;
    if $|G| \geq C$ **then**
        PCH send group assignment notification;
    else
        send task drop notification to v^t;

send out a notification to TV and the selected SVs in G, notifying them of the formation of the group. However, if $H \leq T_{th}$, then the PCH removes the node with the largest M_G (metric calculated based on Equation 4.5 with respect to G) or slowest task computation time, whichever can increase $\hat{T}_G^{stay}$ or decrease $\hat{T}_G^{task}$ the most. Lines 13-14 are repeated until a G with $H \geq T_{th}$ and size larger than the minimum allowed group size C is obtained. If no valid group can be found, the task is dropped and a notification is sent to the requester.

4.5 Application Example and Evaluations

4.5.1 Application Example

As mentioned in the previous section, task completion time prediction is application specific. We consider a general distributed computation task [116] that requires multiple vehicles to cooperate by contributing combined resources of sensor data/local information, computing power, local decisions, etc, where examples are safety related applications [118] and on-board intelligence [119, 116]. In this section, we describe and evaluate an example on-board intelligence application - distributed learning across a vehicular group.

We assume a generalized distributed learning model, similar to the Federated Learning (FL) framework introduced in [126], but we do not address privacy concerns as FL does. In our application example, every car manufacturer has a own centralized server (CS) that can communicate to its manufactured vehicles through a Vehicle to Infrastructure (V2I) protocol. At each round of training, a CS selects a vehicle to initiate the process by sending it a task assignment. Following our proposed ToG algorithm, the selected vehicle will then submit the task request to its PCH and wait for a task group assignment. When the task is successfully completed, the task vehicle will send a task completion notification to its PCH and send model updates to its CS.

The task computation procedure for a given distributed learning task can be decomposed into four major stages:

- *1) Data Sharing:* for each task, a task group member shares its local data to other members, which are selected based on the task requirement with a fixed required length.

- *2) Task Computation:* each task group member collects other members' shared data, which totals to $|G|$ pieces of data (including its own piece). Once $|G|$ pieces of data are obtained, each member calculates and make predictions for the collected data set.

- *3) Results Sharing:* Once computation is done, each task group member multicasts its results with fixed required length to other group members.

- *4) Local Training:* Upon receiving computation results from other group members, a vehicle updates its local model by adding the unlabeled data points to its training set if the majority of the task group agree on the label[3].

After this process is completed, the TV syncs its update with the CS.

Let L_{Data} denote the fixed length of data needed to be exchanged by one task group member for a given task. Similarly, L_{Result} denotes the fixed length of result needed to be send out by one task group member after computing the task. As different models of vehicles may have different computing capability, we use operations per second κ to represent the computation rate of a given vehicle, and O_{Data} to represent total operations needed for given length of data L_{Data}. As the fourth stage – *Local Training* can be completed by a vehicle itself, it does not require stable connections among task group members. Hence, the task completion time as:

$$\hat{T}_G^{task} = \frac{|G| \times L_{Data}}{r_{min}^G} + \frac{|G| \times O_{Data}}{\kappa_{min}^G} + \frac{|G| \times L_{Result}}{r_{min}^G} \tag{4.9}$$

where $|G|$ represents the size of group G, r_{min}^G represents the minimum transmission rate

[3]Averaging local models may balance their contributions to produce an accurate joint model, though this is not guaranteed. Instead, we use tri-training [127], a classic method that reduces prediction bias on unlabeled data by using the agreement of multiple independently trained models.

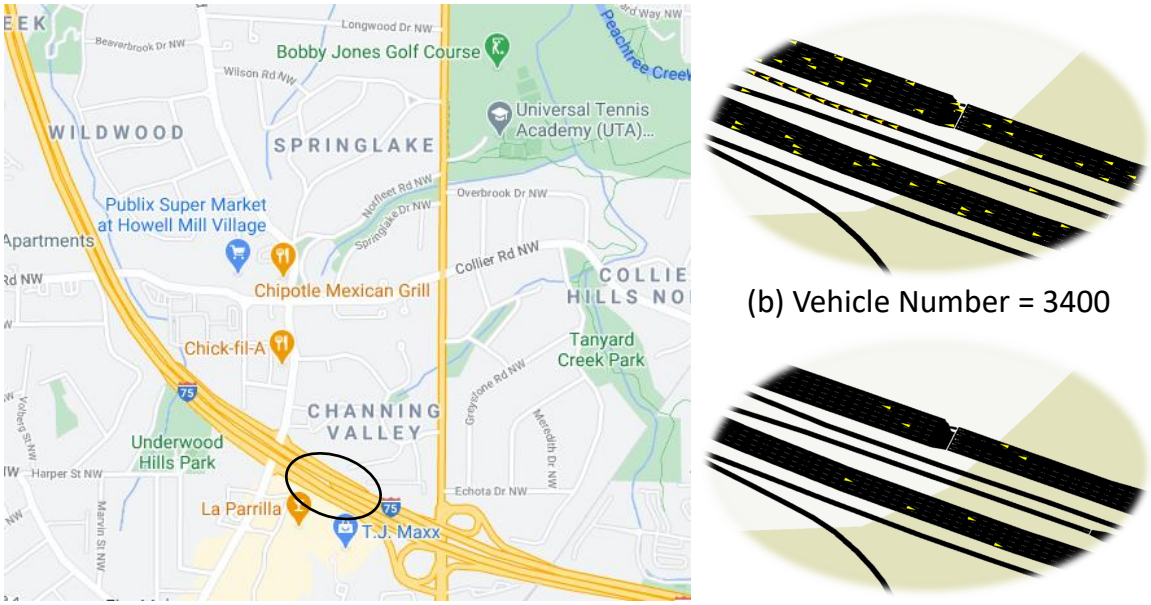


Figure 4.2: (a) 3.5 mile highway section. (b), (c) are captured traffic images on the same 120m highway section. Each small yellow triangle represents a vehicle.

among vehicles in G, and κ_{min}^{G} represents the minimum computation rate of the vehicles in G. By using the minimum transmission rate to estimate the data and result transmission times, we obtain an upper bound, which can be considered to compensate for additional time needed because of packet delays and losses.

By applying the above task completion time estimation scheme, we are able to follow Algorithm 5 to form groups tailored for distributed learning. An extensive simulation study based on this type of application and making use of real world maps is provided in the following subsections.

4.5.2 Simulation Set-Up

We implemented a prototype of ToG using C++ and Python, which can simulate not only different real maps but also different traffic scenarios by updating a small set of system parameters. It is built on top of Veins [95] which provides a comprehensive suite of models of IEEE 802.11p, DSRC/WAVE and obstacle shadowing. We add additional layers to simulate the communication between remote servers and TVs. TraCI from SUMO [91] is used

to control the mobility of vehicles. About 4000 lines of code are written for the prototype. Real highway, branches and intersections are obtained from OpenStreetMap (OSM) [112]) with manual corrections referenced from Google Satellite.

Vehicles can communicate with the centralized servers and nearby vehicles through V2I and V2V, respectively. Figure 4.2(a) depicts a major highway section in Atlanta. Its main road is $\sim$3.5 miles long with 6 traffic lights controlling the connections to the branches, and 5 lanes in each major road of each direction. We simulate the example application as described in Section 4.5.1, where available vehicles are selected as TVs and receive a computation task assignment at random times. Upon receiving a task assignment, a vehicle submits a task request to its PCH asking for a task group formation.

The actual $\hat{T}_G^{Task}$ of a task consists of the actual communication cost and the actual task computation time. Communication cost can be well simulated in the current environment, but the task computation time is quite application specific and it can be affected by various factors, e.g., concurrent background processes, RAM size, etc. Therefore, to better evaluate the effectiveness of our proposed approach, we use two normal distributions $K \sim N(\mu_\kappa, \sigma_\kappa^2)$ and $T \sim N(\mu_\tau, \sigma_\tau^2)$ with adjustable mean and variance to cover a wide range of possible task computation times. As different vehicles may be equipped with different rated chips, at the initial stage, each vehicle entering the simulated area is assigned a computation capability rate κ (FLOPs), which follows the $K \sim N(\mu_\kappa, \sigma_\kappa^2)$ distribution. However, even if every vehicle has the same hardware, the actual task computation time is determined by various factors, and cannot be predicted precisely based on the rated computation capability. Thus, we use another normal distribution $T \sim N(\mu_\tau, \sigma_\tau^2)$ to model these variations in computation time. Therefore, the actual task computation time is the sum of the rated computation time chosen from the K distribution and the computation time variation chosen from the T distribution. To simulate different stay times, we included 6 categories of vehicles and varied the network flow as well as vehicle density by controlling the area throughput. For example, both Figure 4.2(b) and Figure 4.2(c) depict the same

120m segment in the simulated highway with different average numbers of vehicles, where the stay-time would be very different.

4.5.3 Evaluation of ToG Algorithm

For each assigned task, we defined three possible end states: completed, failed, and dropped. A task was counted as completed if the TV received computation results from all group members; alternatively, a task was considered as failed if the TV did not receive computation results from all group members before the assigned group broke apart; finally, if a PCH tried to form a task group but could not find any suitable combination (Equation 4.8), the task was counted as dropped. Note that there were situations that a PCH or TV exited the simulated area while executing their task. As this was caused by the limited simulation area instead of failure of the algorithm, we ignored these cases in the evaluation results. The following metrics are used in our evaluation:

- *Task Execution Rate (TER):* The percentage of completed tasks over the number of completed, failed and dropped tasks. The higher the TER is, the higher the ratio of completed tasks versus task group formation attempts can be obtained, which is one of the ultimate goals of ToC.

- *Average Group Size (AGS):* The average group size of completed tasks. Since larger groups typically produce higher quality results for the cooperative tasks we are interested in, one of our goals is to maximize group size. Since task group size of failed tasks and dropped tasks give no indication on the quality of the formed groups, only completed tasks were counted in this metric.

Unless otherwise noted, the following parameters were used as default setting in all experiments: number of vehicles $= 1800$, $\mu_\kappa = 1.3$ TFLOPs[4], $\sigma_\kappa = 0.1$ TFLOPs, $\mu_\tau = 0.3$

[4]We set the mean computation rate as 1.3 TFLOPs based on the recently released Nvidia Drive AGX chip [120] designed for Level 2 as well as Level 3 autonomous vehicles. This is just for an example demonstration, the proposed algorithm does not rely on any specific hardware.

Table 4.1: Task Execution Rate (TER), Task Completion Rate (TCR), and Average Group Size (AGS) for Different Ways of Choosing Safety Margin T_{th}. In each cell of the table, TER, TCR, AGS are presented.

(a) Scenario A: T_{th} as a Percentage of Predicted Task Completion Time T_G^{Task}

	5%	10%	15%	20%	25%	30%	35%
$\sigma_\tau = 0.2$	83.8,84.6,7.2	89.4,89.8,6.7	91.3,92.1,6.6	92.0,93.2,5.9	89.7,94.1,5.0	88.8,94.9,4.5	87.2,95.0,3.8
$\sigma_\tau = 0.4$	78.5,79.2,7.0	85.9,86.6,6.8	86.3,87.0,6.5	88.7,89.4,5.8	89.4,90.8,4.9	86.2,91.4,4.5	83.5,92.1,4.0
$\sigma_\tau = 0.6$	71.6,71.9,6.8	74.2,76.5,6.7	77.6,80.3,6.4	81.4,82.1,5.9	84.0,84.9,4.8	85.3,88.7,4.4	79.8,89.1,3.9
$\sigma_\tau = 0.8$	58.0,58.1,6.7	60.8,61.3,6.5	63.6,64.9,6.1	66.6,69.3,5.7	70.4,74.8,4.6	78.7,85.6,4.3	85.1,88.0,4.1

(b) Scenario B: T_{th} as a Linear Function of Computation Time Variance σ_τ

	$0.75\sigma_\tau$	$1.0\sigma_\tau$	$1.25\sigma_\tau$	$1.5\sigma_\tau$	$1.75\sigma_\tau$	$2.0\sigma_\tau$	$2.25\sigma_\tau$
$\sigma_\tau = 0.2$	85.6,85.9,7.0	90.8,91.5,6.6	92.0,92.8,6.5	92.3,93.1,6.4	92.5,93.5,6.2	92.6,94.0,5.9	90.8,94.6,5.6
$\sigma_\tau = 0.4$	86.0,86.4,6.9	91.6,91.8,6.5	91.9,92.7,6.4	92.1,92.9,6.2	92.4,93.3,5.9	92.2,93.8,5.7	90.2,94.3,5.3
$\sigma_\tau = 0.6$	86.7,87.0,6.2	90.3,90.5,6.1	90.9,92.4,6.1	90.4,92.8,5.7	90.1,93.4,5.6	89.8,93.9,5.3	88.5,94.1,5.1
$\sigma_\tau = 0.8$	87.2,88.6,4.1	88.5,89.0,4.8	86.3,89.2,5.4	82.4,89.6,5.6	76.8,89.9,4.8	70.2,90.0,4.5	62.7,90.0,4.1

min, $\sigma_\tau = 0.2$ min, $O_{Data} = 135$T operations, and beacon message frequency = 10Hz. The communication range among vehicles was set to $300m$ based on NHTSA's proposed rule [114]. Natural packet loss and delay were simulated such that messages were randomly dropped at receiving vehicles with a drop rate of 10% and packets were randomly delayed within the range of 50ms - 500ms. The minimum size of a task group is 2.

Choosing safety margin

We first evaluated how the safety margin T_{th} affects ToG's performance, to determine a suitable way to configure T_{th}. Recall that H is the difference between estimated group stay time and task completion time. Intuitively, a safety margin tolerates the gap between H and the actual difference of these quantities. Thus, one possibility is to make T_{th} a certain percentage of $\hat{T}_G^{Task}$ (Scenario A). However, a good T_{th} should also create a good separation between the distributions of task completion time and vehicle stay time. Then, the variance of task completion time should play an important role in choosing a proper safety margin. Thus, a second possibility is to make T_{th} proportional to σ_τ, the major variance factor in the actual T_G^{Task} (Scenario B).

Based on this, we did a comparison experiment where we varied T_{th} as both a percentage of the predicted task completion time with 5% increments from 5% to 35%, and as multiples of σ_τ with 0.25 factor increments from $0.75\sigma_\tau$ to $2.25\sigma_\tau$. We ran simulations with the default setting for both scenarios, and varied the σ_τ from 0.2 to 0.8 with 0.2 increments for simulating different task computation time distributions. Thirty simulation runs were done for each parameter combination, where 200 computation tasks in total were distributed and tracked in each run. Every simulation run ends when all distributed computation tasks end in one of the three states (completed, failed, dropped) or the corresponding TV exits the map. Since TER may be heavily affected by large number of dropped tasks if there are no suitable members to form the group, we introduce a new metric to help choose T_{th}: Task Completion Rate (TCR). TCR is the percentage of completed tasks over

the number of completed and failed tasks, where dropped tasks are not counted.

The results are reported in Table 4.1. Table 4.1(a) shows the Scenario A result for each combination of T_{th} and σ_τ in the format of (TER, TCR, AGS). Not surprisingly, for the same type of task computation time distribution (same σ_τ), TCR increases and AGS decreases as T_{th} grows. This is because a longer T_{th} increases the deviation that can be tolerated from the predicted H value. Thus, intuitively, the larger the T_{th}, the higher the TCR should be obtained. Note that different from TCR, TER drops as T_{th} becomes too high, because as T_{th} grows, PCHs may not be able to find enough suitable PCMs to form groups, thereby causing the drop in TER. Also, as σ_τ increases, the overall performance drops. This is due to the fact that, the distribution variance of the task computation time is not considered in this way of choosing T_{th}. The predicted $\hat{T}_G^{Task}$ is solely based on estimated communication cost among task group members and rated task computation time (see Equation 4.9). Therefore, for larger σ_τ, higher T_{th} performs better, while for smaller σ_τ, lower T_{th} leads to better performance.

Table 4.1(b) shows the result of Scenario B, in the same format as in Table 4.1(a). As opposed to Scenario A, we see that making the safety margin proportional to σ_τ makes the results fairly consistent as σ_τ is varied. For any given T_{th} value, both TER and TCR are fairly stable for σ_τ in the range [0.2, 0.6]. It is only when σ_τ becomes 0.8 and the safety margin is large, that we see a significant drop-off in TER, which is the ultimate metric of interest. We note that, when $\sigma_\tau = 0.8$, the variance becomes unreasonably large for this simulation scenario and the poor performance is largely due to failed tasks arising from group members exiting the map before task completion.

This comparison shows that, if the value of σ_τ is known, T_{th} in the range of $[1.0\sigma_\tau, 2.0\sigma_\tau]$ give consistently good results on the ultimate metric, task execution rate, while also achieving good group sizes. Obviously, knowing σ_τ means that the task completion time distributions must be characterized for the tasks being executed. We briefly discuss ways in which this could be done in Conclusion. The Scenario A results show that, without knowledge of

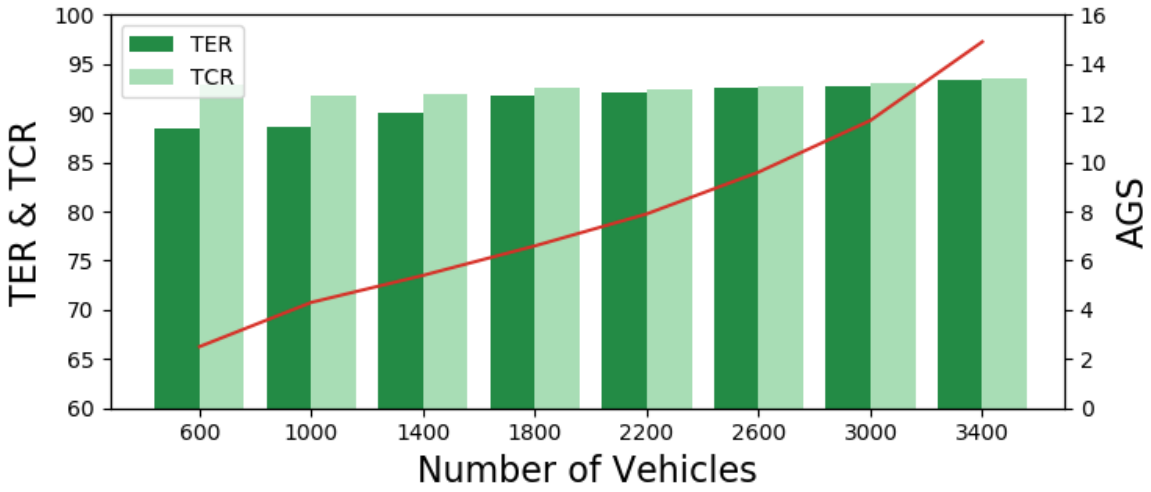


Figure 4.3: ToG Performance vs. Number of Vehicles

σ_τ, a larger safety margin might be needed to tolerate the possible range of σ_τ values and both TER and average group size will be somewhat lower than for Scenario B but good performance is still achieved.

Based on this discussion, in the remainder of the simulations, we assume σ_τ is known and we set $T_{th} = 1.25\sigma_\tau$ and $\sigma_\tau = 2$ as a representative scenario for further evaluations.

Impact of the number of vehicles

We also evaluated ToG's performance versus the number of vehicles, as this parameter has a large impact on the vehicle stay time distribution. For the number of vehicles ranging from 600 to 3400 in increments of 400, we repeated the simulation 30 times for each case. The results are reported in Figure 4.3. From the figure, we can see that as the number of vehicles increases, TER becomes closer to TCR and AGS increases from 2.6 to 14.9. Though when the number of vehicles is very small and vehicles move very freely, ToG is still able to achieve a TER above 87%, which shows the strong potential of ToG's estimation of stay time, making it robust to various traffic scenarios that produce widely different stay time distributions.

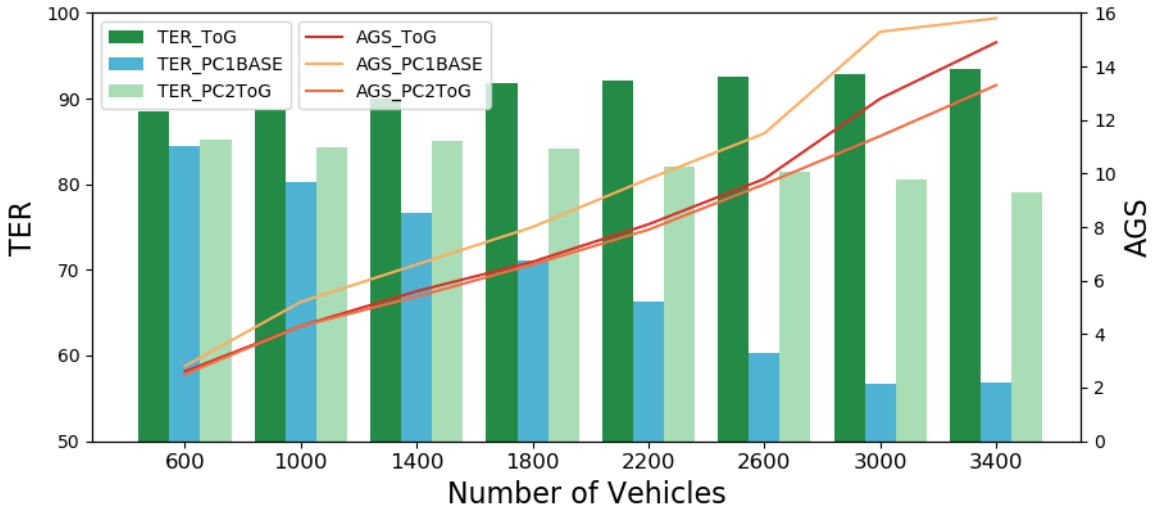


Figure 4.4: Performance of 3 Approaches vs. Number of Vehicles

Table 4.2: ToG Performance vs. Packet Delay

Delay (ms)	[50,250)	[50,450)	[50,650)	[50,850)	[50,1050)
TER	92.5	92.1	91.4	90.2	87.3
TCR	92.8	92.6	92.0	90.9	88.5
AGS	6.7	6.6	6.3	6.1	5.4

Impact of communication delay

As the communication environment plays an important role in affecting group stability, we also evaluated performance over a wide range of packet delays[5], which was varied from [50,250) ms to [50, 1050) ms with 200 ms increments on the maximum delay. Thirty simulation runs were conducted for each case and the results are reported in Table 4.2. From the table, we can see that as the packet delay range expands, TER, TCT and AGS show only small decreases until the packet delay reaches 1050 ms. Even in the range of [50,1050) ms, our proposed method is still able to achieve 87.3% TER with 5.4 AGS, which is still good performance. Thanks to the safety margin design and loosely bounded task completion time estimation, ToG shows good tolerance to communication delays while maintaining high TER, TCR and AGS.

[5]Very long packet delay emulates the impact of packet loss.

4.5.4 Comparison of ToG Algorithm with Other Approaches

To better illustrate the benefits of the ToG Algorithm and how each stage affects the final performance, we compared its performance against two baseline approaches:

- 1) *PC1Base*: an approach that adopts the same first-stage parent clustering scheme as our proposed ToG scheme but uses a baseline second-stage task group formation methodology, which selects task group members that are within communication range of the TV and from which the PCH has received at least one beacon message update in the most recent 5 cycles.

- 2) *PC2ToG*: an approach that uses a well cited clustering scheme [128] in the first stage and adopts the same task group formation method as the ToG Algorithm in the second stage.

By choosing these alternative approaches, we hope to, in part, determine the relative impacts of the first stage (parent clustering) and second stage (task group formation) of our ToG approach. Figure 4.4 depicts the performance of the three algorithms versus different numbers of vehicles (from 600 to 3400 with increments of 400).

We first compare the performance of ToG to that of PC1Base. It is observed that the TER of PC1Base drops quickly (from 84.4% to 56.8%) as the number of vehicles increases. Though PC1Base and ToG adopt the same parent clustering scheme, in the task group formation stage, ToG considers the distribution relationship between $\hat{T}_G^{Stay}$ and $\hat{T}_G^{Task}$, and selects PCMs with a better $\hat{T}_G^{Stay}$, $\hat{T}_G^{Task}$ distribution separation (through T_{th}) to form a task group, while PC1Base only considers distance and signal metrics and ignores the factors of different vehicles' computing capabilities as well as stay times of different groups. With PC1Base, as the number of vehicles increases, more PCMs within communication range and with good connection are not able to finish task computation before separation, causing the large decrease in TER. Nevertheless, as is expected, the AGS achieved by ToG is slightly smaller than that achieved by PC1Base, because ToG has more constraints in se-

lecting task group members as a trade-off for better TER. Overall, this comparison shows the clear benefits of our task-oriented group formation in successfully completing a much higher percentage of tasks while achieving close to the same average group size as an approach that does not factor in the task requirements when choosing a computation group.

Next, we compare the performances of PC1Base and PC2ToG. Similar to PC1Base, PC2ToG's TER decreases as the number of vehicles increases, but not by as much (only from 85.2% to 79.3%). Recall that PC2ToG adopts the same task group formation scheme as ToG, but uses a different parent clustering scheme. This shows that, while both ToG's clustering scheme and its task group formation scheme improve the task execution rate, the task group formation scheme is the more important factor and using it with other parent clustering schemes still provides substantial benefits.

In the end, if we put an eye on performance among all three algorithms, we observe that only ToG's TER increases as the number of vehicles increases. This is because our parent clustering scheme both clusters vehicles that tend to stay longer together, which provides good candidates for second stage task group formation, and is robust across different vehicle densities. Then, in the second stage, only PCMs with higher potential of completing tasks together before losing connection are selected as task group members. Besides, it always tries to find the largest possible group to achieve better model training results. Therefore, we see that AGS grows as the number of vehicles increases. Moreover, although PC2ToG results in lower TER than ToG, it has better TER performance than PC1Base, and this shows that our second-stage task-oriented group formation algorithm can be combined with different parent clustering schemes to improve the success rate of tasks completing.

4.6 Chapter Summary

We have demonstrated that our ToG approach has great potential in achieving high TER while maximizing AGS, though it requires some prior knowledge of application-specific

task completion time distributions. The estimation and statistical modeling of task completion time is an interesting area. Due to time limits, we were not able to address that aspect in this thesis. However, we will definitely include that part in our future work (refer to Section 7.2 for further details).

CHAPTER 5

MULTIVTRAIN: COLLABORATIVE MULTI-VIEW ACTIVE LEARNING FOR SEGMENTATION IN CONNECTED VEHICLES

Since the information security problem and task-oriented group formation challenge are addressed in the previous two chapters, in this chapter, we move to the next step of connecting online learning and active learning to design an online active learning system for connected vehicles, where vehicles jointly increase training data availability while requiring much smaller data storage as well as improve model accuracy with faster model updates without the support of human annotation and remote data centers. Because semantic segmentation is a harder task than classification, and has more use-cases for vehicles, in this chapter, we use semantic segmentation task as an example to design the framework, but the framework we proposed is not limited to semantic segmentation only.

To overcome the barrier of insufficient high quality training data covering a complex range of vehicular scenarios, 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.

5.1 Introduction

Semantic segmentation is of great significance for autonomous vehicles to understand the environment [3], 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 [129, 130, 65]. 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 [80]. Prior works have demonstrated the great potential active learning can add to the training performance as well as efficiency [84, 83, 131, 77]. 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 [121]. 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 work, 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.

5.2 System Model and Preliminaries

In this section, we first present our system model in Section 5.2.1. As depth-fused images play an important role in our proposed approach, the pre-processing and fusion procedures we adopt are introduced in Section 5.2.2.

5.2.1 System Model

As shown in Figure 5.1, 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 Figure 5.1. 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 Figure 5.1. 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.

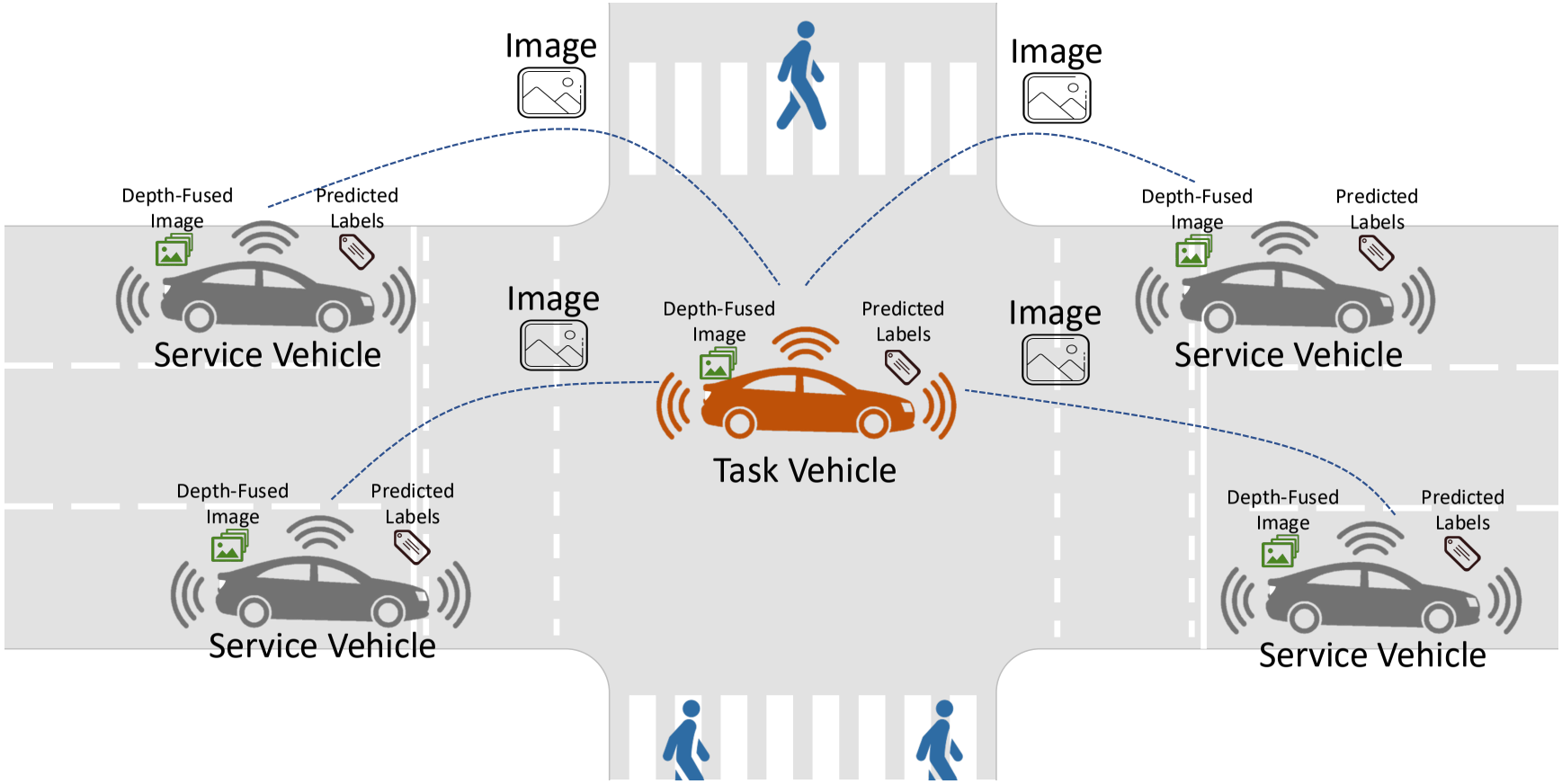


Figure 5.1: MultiVTrain System Model Overview

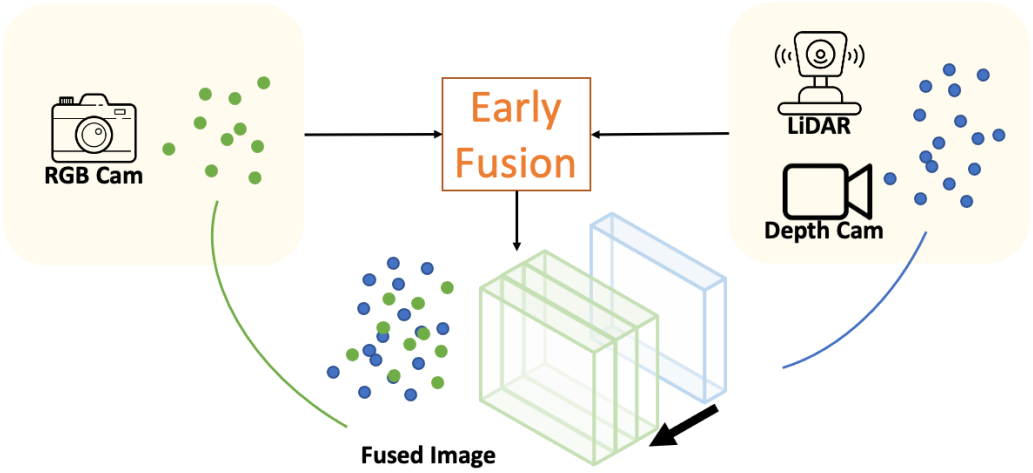


Figure 5.2: Depth-Fused Image

5.2.2 Depth-Fused Images

Sensor fusion is used to aggregate correlated multi-view images from nearby vehicles for pseudo labeling. As shown in Figure **??**, 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 [132], 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 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 cal-

ibration, the point cloud can be projected to the image pixel array according to:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{5.1}$$

where K, R, and t denote the intrinsic matrix, rotation matrix, and translation vector of the RGB camera, respectively [132]. By applying Equation 5.1 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.

5.3 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 [133] and 3D shape belief transfer proposed in [134], 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] All resulting points $[u, v]$ that fall outside the boundary of the image plane are discarded.

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 Figure 5.3 (a) and Figure 5.3 (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;

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 5.2.2, 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 5.2.1, 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:

$$[\phi_{j,n}(\mathcal{I}_i^t \leftarrow \mathcal{I}_j^t)|\epsilon] = [\phi_{j,n}(\mathcal{I}_j^t)|D_j^t]K_j^{-1} \begin{bmatrix} R_j & t_j \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} R_i & t_i \\ 0 & 1 \end{bmatrix} K_i \tag{5.2}$$

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 Figure 5.3 (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.

MPT results improvement

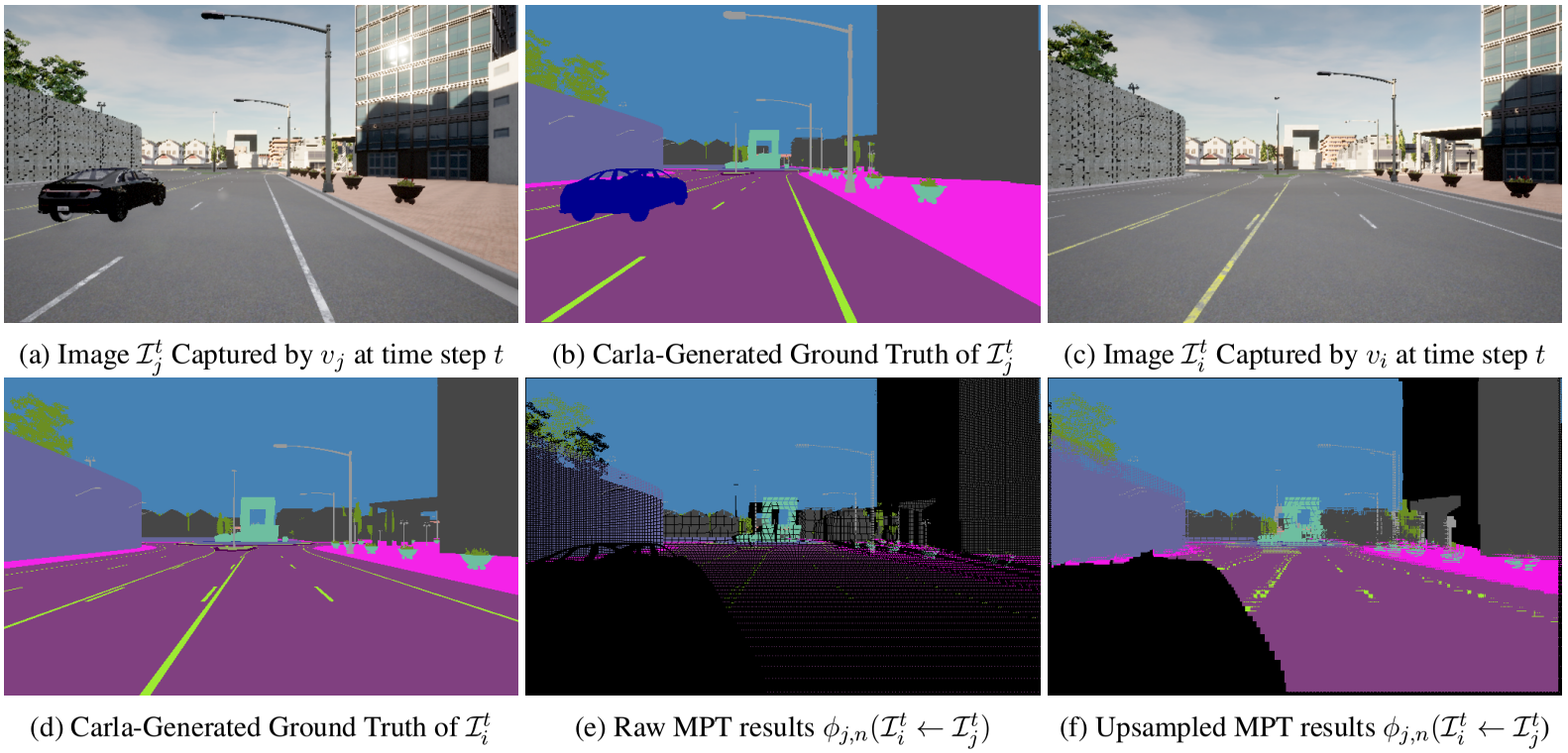
As we can see from the figure that the depth information is missing at certain pixel locations (black in Figure 5.3 (e)), this is due to either out of range objects or the sparsity of captured point clouds. If the point clouds are too sparse, the benefit of this scheme will be deteriorated, because there would be little information to correctly correlate pixel locations in diverse views. We follow the interpolation method introduced in [135] to up-sample 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 $+\infty$.

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.

5.4 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 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



(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 $\phi_{j,n}(\mathcal{I}_i^t \leftarrow \mathcal{I}_j^t)$

Figure 5.3: Example Scenario of Multi-View Prediction Transfer (MPT)

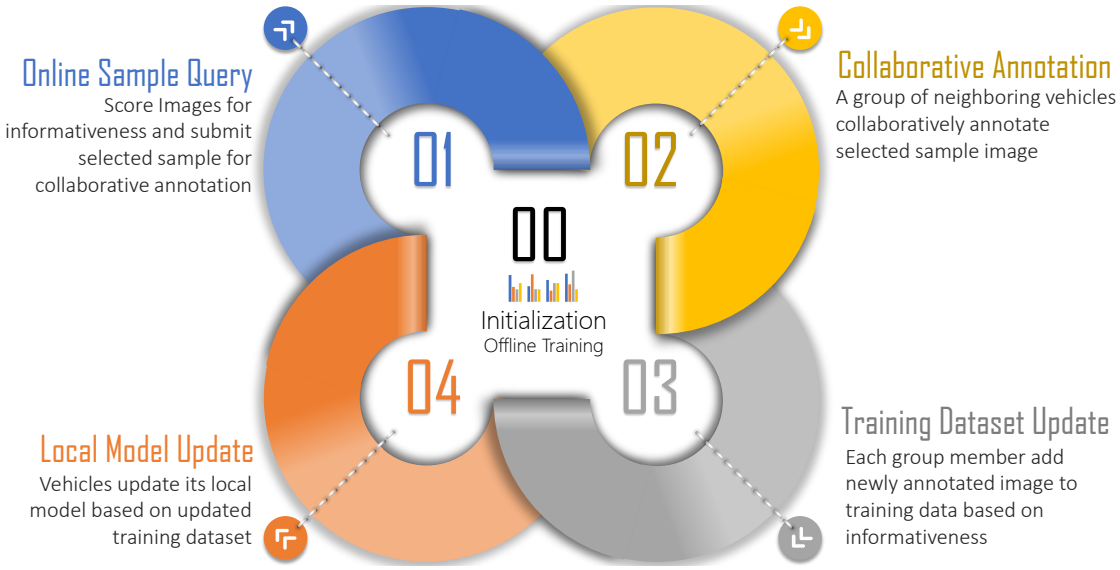


Figure 5.4: Online Active Learning Framework Loop

is illustrated in Figure 5.4.

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.

5.4.1 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 [136]. We use the standardized sum of pixel-wise cross entropy to measure the segmentation loss:

$$\mathcal{L}_{ce} = -\frac{1}{N} \sum_{n=1}^{N} \sum_{c=1}^{C} y_{n,c} \log(\phi_{n,c}) \tag{5.3}$$

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.

5.4.2 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* [80]. 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 [137]. Thus, the informativeness scoring $\mathcal{S}$ of an image $\mathcal{I}$ with N pixels is calculated by cross-entropy as:

$$\mathcal{S}(\mathcal{I}) = \frac{1}{N} \sum_{n=1}^{N} \mathcal{H}(\phi_n) = -\frac{1}{N} \sum_{n=1}^{N} \sum_{c=1}^{C} \phi_{n,c} \log(\phi_{n,c}) \tag{5.4}$$

$\mathcal{H}(\phi_n)$ 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 5.4.4

5.4.3 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 Section 5.2.1, 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 5.3, 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 5.1, the TV ($v_i \in G$) will form a group G with

Algorithm 4: Depth-Based Weighted Voting Scheme for Pseudo Label Integration

task vehicle: $v_i \in G$, service vehicle: $v_j \in G$;
while v_i *selects a valid sample image* $\mathcal{I}_i^t$ **do**
  v_i broadcast the corresponding depth-fused image $\mathcal{D}_i^t$ and prediction $\phi_i(\mathcal{I}_i^t)$;
  for *all available* v_j *receives* $\mathcal{D}_i^t$, $\phi_i(\mathcal{I}_i^t)$ **do**
    computes prediction $\phi_j(\mathcal{I}_i^t)$ using v_j's local model;
    apply MPT scheme (Equation 5.2) and obtain $\phi_j(\mathcal{I}_i^t \leftarrow \mathcal{I}_j^t)$;
    for *each pixel* n *in* $\mathcal{I}_i^t$ **do**
      if $(n : \mathcal{I}_i^t \leftarrow \mathcal{I}_j^t)$ *exist in* $\mathcal{I}_i^t$ **then**
        obtain $\mathcal{L}_{j,n}(\mathcal{I}_i^t)$ based on Equation 5.5
      else
        $\mathcal{L}_{j,n}(\mathcal{I}_i^t) = \phi_{j,n}(\mathcal{I}_i^t)$
    broadcast $\mathcal{L}_{j,n}(\mathcal{I}_i^t)$ to other group members;
  for *all members in* G **do**
    take v_i's $\phi_i(\mathcal{I}_i^t)$ as $\mathcal{L}_{i,n}(\mathcal{I}_i^t)$;
    for *each pixel* n *in* $\mathcal{I}_i^t$ **do**
      $\mathcal{L}_{G,n}(\mathcal{I}_i^t) = \frac{1}{|G|} \sum_{v_k \in G}^{|G|} \mathcal{L}_{k,n}(\mathcal{I}_i^t)$
  go back to the start of WHILE loop;

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 5.3) 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:

$$\begin{aligned} \mathcal{L}_{j,n}(\mathcal{I}_i^t) &= \gamma_{j,n}\phi_{j,n}(\mathcal{I}_i^t \leftarrow \mathcal{I}_j^t) + (1-\gamma_{j,n})\phi_{j,n}(\mathcal{I}_i^t)), \\ \gamma_{j,n} &= \frac{\frac{1}{2} - \frac{1}{2}\sin(\frac{|\alpha_{j,n}^t| - |\alpha_{i,n}^t|}{\bar{\alpha}})}{1 + e^{\frac{d_{j,n}^t - d_{i,n}^t}{\bar{d}}}} \end{aligned} \tag{5.5}$$

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

[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.

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.

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).

5.4.4 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 [138]. To be more specific, we adopt the naive rehearsal method [139], 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. Due to GPU limit, we use $(8, 8)$ split instead. Though the $(64/64)$ setup listed in the original paper is more ideal, the $1:1$ ratio between new data and the rehearsal data in each batch size is more important.

5.5 Simulation Setup and Data Collection

5.5.1 Experiment Setup

The evaluation is performed using the Carla simulator [98], 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.

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[3], and an RGB camera were attached to each vehicle and calibrated in the same way, so that with the application messages as described in Section 5.2.1, a vehicle was able to calculate neighboring vehicles'

[3] 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.

Table 5.1: 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)

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.

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 pro-

posed 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 Section 5.5.2. 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.

5.5.2 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.

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 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 5.4.4. 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".

Training for Baselines

We compare our approach with different baseline algorithms by adapting two existing supervised learning methods, PSP [140] and DeepLabv3 [141], and the active learning schemes in [142, 2].

- ***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.

- ***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 [142], 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 [2], to obtain ACWA-acpl.

A summary of the training methods and applied training sets for all models is provided in Table 5.1.

5.6 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:

$$\mathrm{IoU} = \frac{n_{cc}}{n_{cc} + \sum_{\eta \neq c}^{C} (n_{\eta c} + n_{c\eta})} \tag{5.6}$$

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 5.2. Ten models with different configurations are tested with the same unseen test set (collected from Town05 as described in Section 5.5.1). Per-class IoUs are shown in the first 13 data columns and the mIoU is provided in the last column.

5.6.1 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;
3. whether it is vulnerable/sensitive to the common forgetting issue that affects some AL approaches.

Table 5.2: 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

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 5.1). 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 5.2, 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.

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 5.2, 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.

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 unseen data and gradually forgets about important information learned in the past. By integrating the rehearsal-C approach [139] 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.

5.6.2 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 5.2, 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.

5.6.3 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.

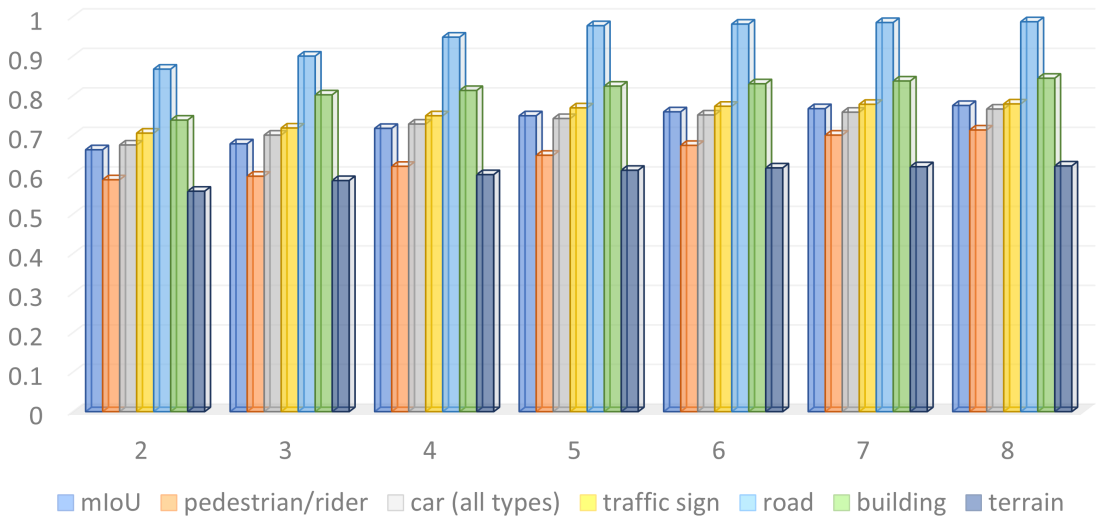


Figure 5.5: Model Accuracy (mIoU, IoU) vs. Group Size

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 Figure 5.5. 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 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.

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

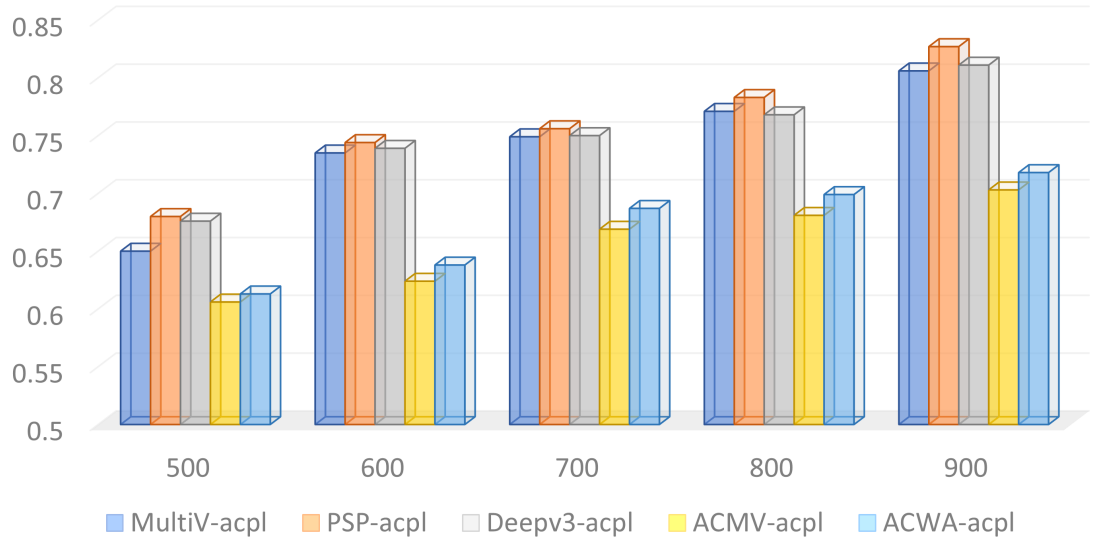


Figure 5.6: Model Accuracy (mIoU) vs. Initial Training Set Size

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 Figure 5.6. 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.

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 time limit, we cannot evaluate these parameters in detail but a short discussion is provided.

- ***Diversity of Views***: as demonstrated in prior computer vision works [134] 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 5.6.1 and Section 5.6.3. However, given the same group size, we were not able to evaluate how the diversity of views affects the results.

- ***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.

5.7 Chapter Summary

In this chapter, 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.

CHAPTER 6

SIM2SCALE: A CO-SIMULATION FRAMEWORK FOR DEVELOPING AND EVALUATING ALGORITHMS FOR CONNECTED AND AUTONOMOUS VEHICLES

As discussed in previous chapters, CAVs have the potential to reduce accidents, enhance the quality of life, and improve traffic/energy efficiency. Despite their conceptual appeal, developing as well as evaluating algorithms/applications in CAVs is a challenge, where the 3D scenario, traffic/mobility, and inter-vehicle communication are all important aspects that should be incorporated. Running experiments in a testbed is costly and cannot reproduce many conditions that a vehicle encounters in the real world. Besides, existing simulators can hardly support co-simulation of all three of the aforementioned aspects. In this thesis, we describe Sim2Scale, a co-simulation framework we developed for CAV algorithms/applications, which integrates 3D scenarios, traffic/mobility, and inter-vehicle communication. Sim2Scale is built on top of the open-sourced libraries of SUMO, Veins, and CARLA, and provides both Python script and GUI interfaces for ease of use. The thesis also describes a collaborative active learning application example and presents a basic evaluation of it using the Sim2Scale framework in order to illustrate the effectiveness and customizability of the framework.

6.1 Introduction

The development as well as deployment of CAVs requires extensive evaluations of developed control, perception, localization, and more advanced algorithms. Directly testing on real road may not only lead to tremendous cost, hardly reproducible testing results, but also serious injuries [143]. Thus, extensive simulations and verification in a realistic simulation environment before proving ground and public road testing is of crucial importance.

Nonetheless, there is a gap between the growing demands and currently available simulation platforms. First, the level of detail in the simulation environment helps ensure the safety of a real-world implementation and reduces algorithm development cost by allowing developers to complete most of the validation in the simulation environment before testing on real roads. Considering sensor sets like camera, LiDAR, and communication protocols like V2V, V2X used in CAVs, it is of significant importance to create a simulation platform that support sensor simulations as realistically as possible, where a realistic 3D scene becomes an underlying requirement. Second, while sensor and 3D scene simulation are essential for perception algorithm development, such simulation platform will be incomplete for the simulation of holistic CAVs operation without being complemented by a realistic mobility model and various traffic scenarios. Third, as mentioned in [144], various distributed computation tasks require not only data sharing but also cooperative computation among vehicles [116], e.g. reinforcement learning based cooperative driving [117], distributed consensus based false information filtering [118], collaborative active learning [145], etc.. Therefore, providing realistic simulation of different communication protocols in vehicular networks such as V2V, V2I, and V2X is also a necessity. Though various simulation platforms like VISSIM [93], Carla [98], SUMO [91], AutoSim [92], Veins [95] exist, they are either not fully open-sourced or cannot simulate different aspects of sensors, mobility/traffic and communication along with each other, showing limitations on evaluating applications affects by all three aspects.

To address the above challenges, in this chapter, we present a co-simulation platform framework to develop, evaluate and validate connected autonomous vehicles technologies, which requires realistic simulation of 3D scene/sensors, mobility as well as traffic, and communication in run time.

6.2 Architecture

In this section, we first present an overview of our Sim2Scale co-simulation framework, as illustrated in Figure 6.1, and then describe each of its building blocks and their realization details.

6.2.1 Framework Overview

As discussed in Section 2.3 Section 6.1, there are various commercial and open-sourced vehicular simulators that support a part of our simulation goal. In this proposed work, three widely used open-sourced simulation platform are adopted and integrated into one combined platform:

- SUMO [91], a highly portable, microscopic and continuous traffic simulation package designed to handle large networks, is selected to model the scalable mobility of vehicles and different traffic scenarios in a given map;
- Veins [95], which provides comprehensive models of IEEE 802.11p and IEEE 1609.4 DSRC/WAVE network layers, including multi-channel operation, QoS channel access, noise and interference effects, etc., is selected to serve the simulation purpose of connectivity among vehicles;
- CARLA [98], which provides open digital assets (urban/suburban layouts, buildings, vehicles, etc.) that were created to support development, training, and validation of autonomous driving systems, is selected to simulate the 3D environment and interactions with various sensor sets such as lidar, camera, radar, etc..

Besides, as CARLA itself is designed for simulating and validating autonomous vehicle applications, and provides flexible architecture, Python APIs, as well as co-simulation capabilities with SUMO, our proposed work follows the architecture of the CARLA suite, and integrates the communication simulation features of Veins into CARLA/SUMO.

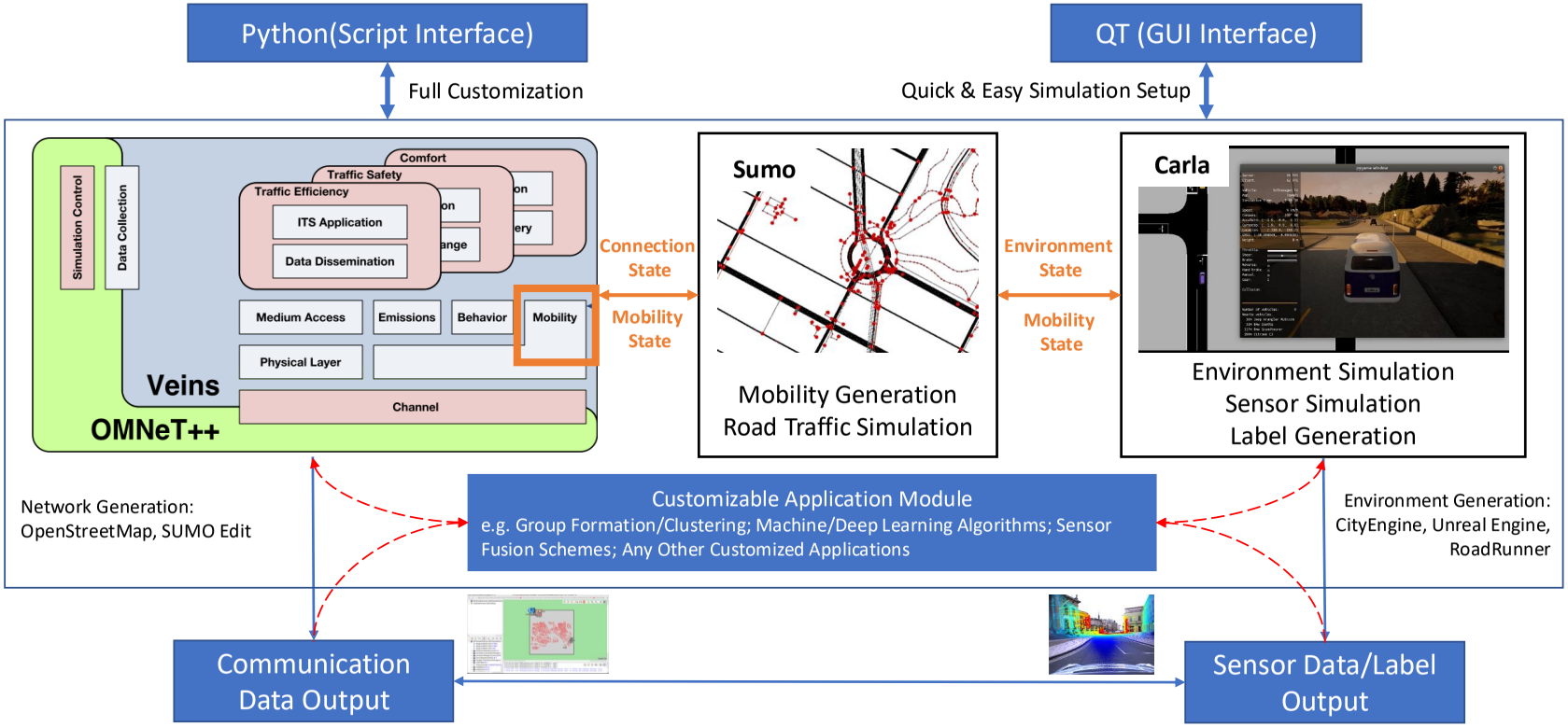


Figure 6.1: Sim2Scale Co-Simulation System Overview

As shown in Figure 6.1, Sim2Scale is a co-simulation framework build on top of SUMO, Veins, and CARLA. SUMO itself serves a central TraCI server (TCP connection), simulating vehicle mobility and traffic. Veins and CARLA, both serve as standalone SUMO TraCI clients, simulating the inter-vehicle communication and 3D environment, respectively. At each time step, Veins and CARLA send communication state and environment state to the SUMO server and wait for the server to reply with the next mobility state. Other application level algorithms such as group formation, sensor data fusion, and deep learning algorithms can be implemented as submodules to the Veins or CARLA clients (refer to Section 6.3 for an illustrative example). For ease of use, two interfaces (Python script and QT GUI) and a data output handler are provided. These allow users to run, customize, record, and access needed data seamlessly.

6.2.2 Interfaces

Sim2Scale provides both Python script and QT GUI interfaces. The Python interface extends from the Python API of CARLA and is fully customizable, whereas the QT GUI interface provides a quicker and easier configuration as well as visualization of certain preloaded parameters, but is less customizable. Details of both interfaces are provided next.

Python Interface

By using the original Python API provided by CARLA, users are able to select maps, change weather, spawn/control vehicles, pedestrians, and other types of moving/static actors. We extend the script interface on top of the existing CARLA APIs by adding configuration support to parameters of SUMO and Veins Suite, e.g., different mobility patterns, traffic densities, sensor usage/installation, sensor data fusion, communication models, etc.

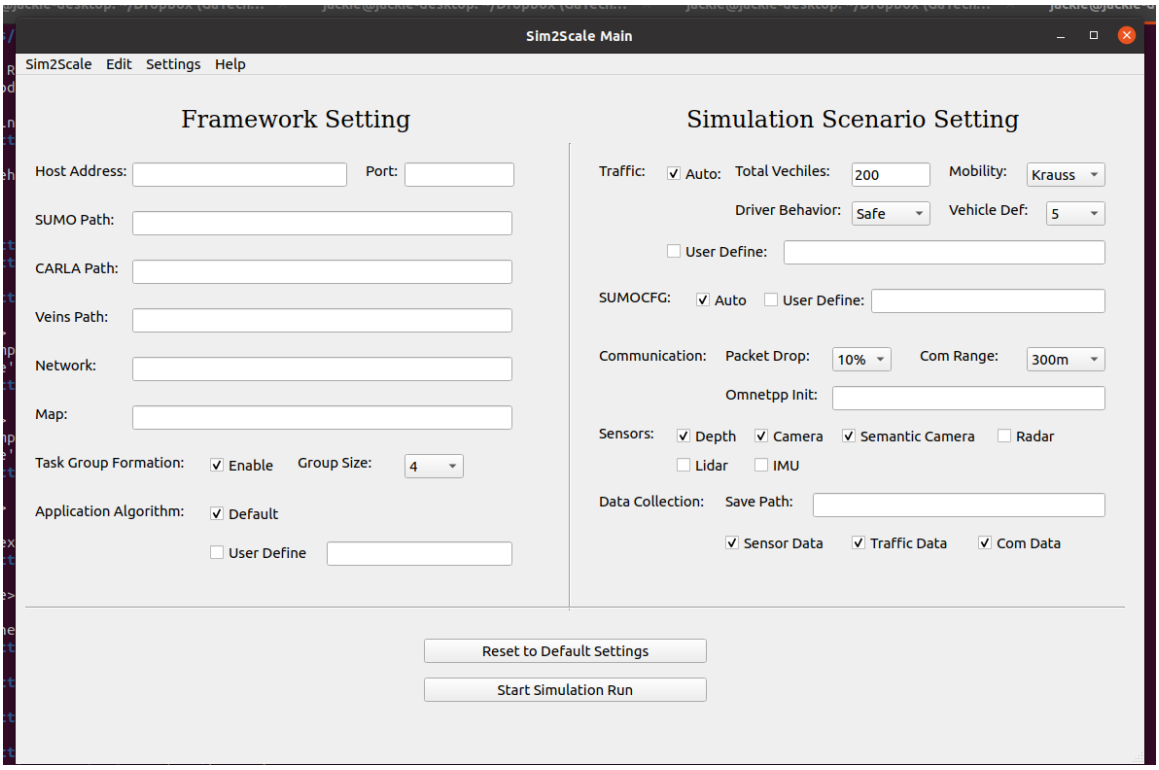


Figure 6.2: Sim2Scale QT GUI Interface

GUI Interface

Though the Python script interface provides better customization support, a GUI interface with visualization is easier and faster for repetitive evaluations. A GUI interface is shown in Figure 6.2, where two sets of configuration parameters are allowed: Framework Setting and Scenario Setting. In Framework Setting, users can quickly configure the environmental path for SUMO server, Veins/Carla clients, and maps/networks with which the simulation is going to run. Any valid map/network obtained from OpenStreetMap [112] with CARLA usable 3D scene is loadable. In Scenario Setting, users can easily obtain reasonable mobility and traffic according to the given map by configuring the total number of vehicles, mobility model, and driver behavior. Moreover, commonly used communication parameters such as packet drop rate and vehicle communication range are configurable through the GUI. For users' convenience, certain commonly used sensors can be selected in the GUI, which causes them to be installed on vehicles based on the KITTI standard [4] and

automatically collect data during the simulation run. Note that in our provided simulation example, a simple task group formation algorithm is included, which serves as a good implementation example for those who want to study applications that require collaboration or clustering among vehicles. Users who do not need this feature can simply untick the "Enable" button to disable this feature.

6.2.3 Traffic & Mobility Simulation

Although CARLA can create traffic around the target area by randomly spawning vehicles and setting them to run in autopilot mode, its built-in vehicle dynamics are modeled based on NVIDIA PhysX Vehicle, which cannot represent vehicle dynamics accurately. To obtain more realistic traffic simulations, SUMO are used to generate traffic in the given map. SUMO adopts the microscopic traffic model, where each vehicle and its dynamics are simulated individually with various extended configurations like driver behavior (impatience), vehicle substructure, etc. Therefore, by generating vehicle dynamics with SUMO's car following model, lane-change model, and junction behavior model, more realistic and reasonable vehicle dynamics are obtained.

6.2.4 Inter-Vehicle Communication Simulation

Advances in communication technologies for CAVs allow vehicles to exchange information through various communication protocols (V2V, V2I, V2X). 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.

Nevertheless, extending Veins suite to Carla in a decoupled way is not an easy task. Although both Veins and Carla support co-simulation with SUMO, Veins is mainly written with C++ with development of Python APIs in a very early stage. Moreover, Veins is built on top of open-source OMNeT++ [146] and SUMO, where road traffic simulation is

performed by SUMO and network simulation is performed by OMNeT++ along with the physical layer modelling toolkit MiXiM. The way Veins and CARLA incorporate SUMO makes it difficult to extend Veins to CARLA. As a result, we abandoned the built-in SUMO launcher from Veins, and developed a new launcher script and clock master to control as well as synchronize the inter-module communication among Veins, SUMO and CARLA. However, all network communication models are directly adopted from Veins. Full capability of the original Veins suite is maintained.

Additionally, a Python-based communication data output wrapper is provided to ease integration with the C++ code. Other modules can access all collected communication data up to time step $t-1$ through the wrapper during run time when the simulator is in time step t.

6.2.5 3D Environment & Sensor Simulation

With the synthetic driving scenarios, we can conduct multiple simulation runs in a deterministic and reproducible fashion. As CARLA has already embedded both urban and suburban synthetic 3D scenes, with realistic variations, we put our focus on the sensor simulation side. First, we standardized the sensor installation position based on the KITTI standard [4]. Then we developed a sensor data collection API to support the different sensor data collection and operation requirements. For example, a camera can be easily configured to collect RGB data/single-color-channel data, with/without segmentation label, with/without coordinate projection, with/without data fusion with other sensors like lidar, save-to-drive or stream to application, etc.. With these sensor APIs, a user can quickly configure sensor settings and get ready to run based on their use case with just a few lines of Python code.

6.2.6 Application Simulation

As shown in Figure 6.1, a decoupled application module is included along with the SUMO, Veins, and Carla modules. During run time, it can access both environment/sensor data and communication data through the data output wrapper and current state variables directly from Veins and CARLA. An example collaborative active learning use case (refer to Section 6.3) has been implemented to demonstrate how to access data from other modules, however, code in this module can be fully replaced by customized scripts.

6.3 Example Scenario

We implemented an example scenario in a synthetic town provided by CARLA, where a baseline collaborative active learning algorithm for image segmentation is simulated following a similar approach to [145, 2]. CARLA's built-in map Town05 is used as our simulation scene. Each vehicle is equipped with a local model, which is a pre-trained segmentation model that is capable of producing coarse segmentation labels in a captured image[1]. A vehicle that is designated as a task vehicle selects a sample image to learn in each round of active learning. Then, it tries to form a task group with neighboring vehicles within its communication range through a simple task group formation scheme described in Algorithm 5 (refer to Section 6.3.2 for details) to collaboratively compute annotation labels in the selected sample image. Lastly, the task vehicle adds the selected sample image to its training data set and updates its local model accordingly (refer to Section 6.3.1 for further details). As the co-simulation including mobility/traffic, network communication, 3D sensor data processing, and application is run on a desktop with a single GPU, we simplified the simulation to include only one task vehicle and evaluated the performance of our proposed Sim2Scale platform by monitoring the events related to the designated task vehicle. However, the framework supports larger scale simulations and more complex tasks,

[1]The labels are from the Cityscapes dataset categories, available at https://www.cityscapes-dataset.com/dataset-overview/class-definitions.

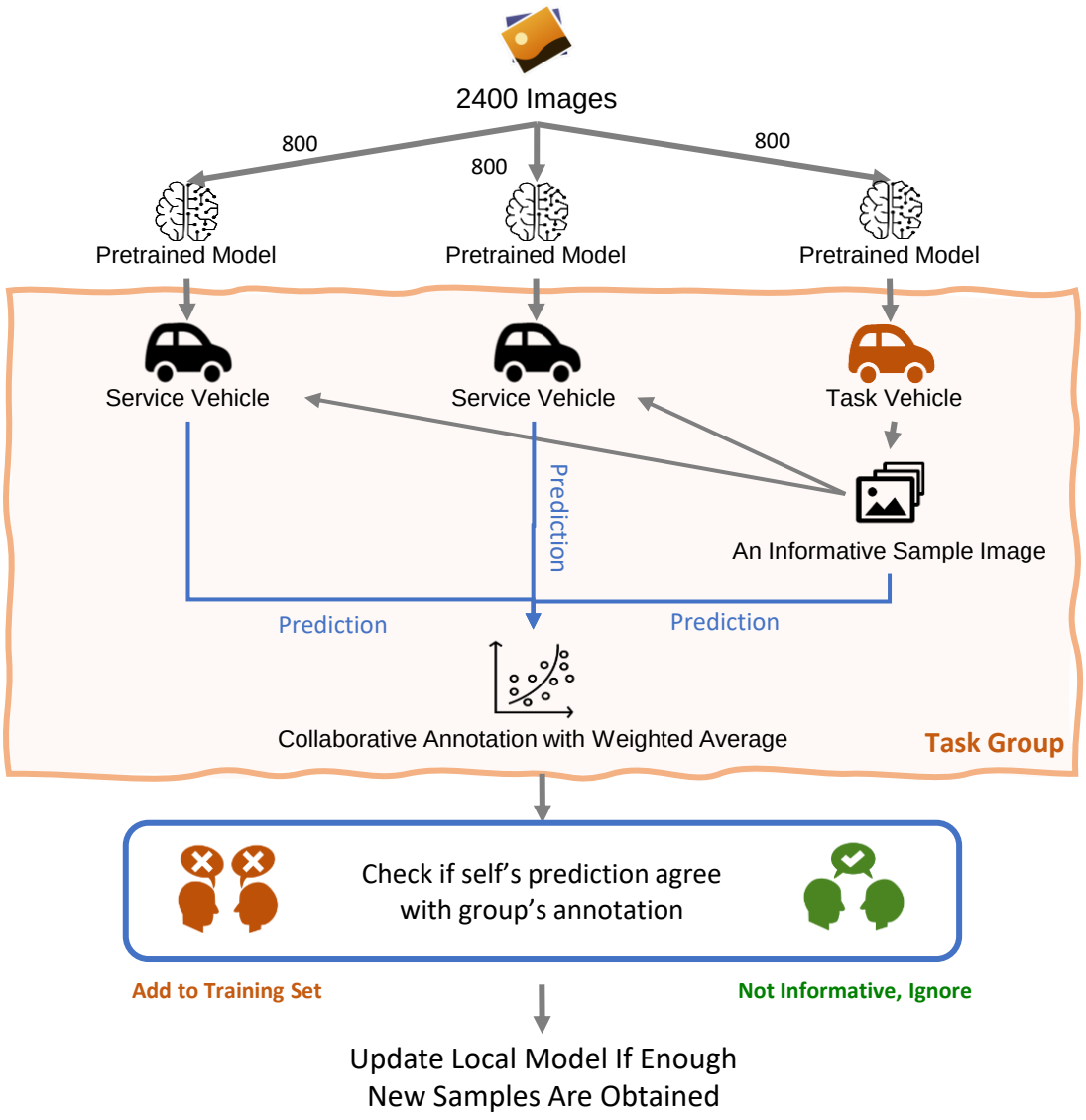


Figure 6.3: Collaborative Image Segmentation Overview

especially with stronger hardware and/or more time for simulation.

6.3.1 Collaborative Active Learning Algorithm

Figure 6.3 provides an overview of the example collaborative active learning algorithm, which is adapted based on a framework used in [145] and [2]. We first run a single vehicle in Town01 and Town07 to collect an image data set to obtain pre-trained models. These towns are different from Town05, which is used for simulation, to ensure the run time data is unseen. A total of 2400 images were collected (we refer to this dataset as TS0), with 1200 images from Town01 and 1200 images from Town07. Each vehicle in the simulation is then equipped with a pre-trained segmentation model, which is trained with 800 randomly selected images from TS0 (a different set of random images for each simulated vehicle). Each pre-trained model is tested on the same test set and obtained a rated model accuracy. Let A_{v_i} denote the rated model accuracy of the local model on vehicle v_i. The model accuracy is calculated by the mean Intersection-over-Union (mIoU), which is a commonly used metric for evaluating the performance of a segmentation model.

During run time, the task vehicle samples an image every 100ms (in simulation time). If the sampled image is considered informative based on cross entropy[2], it selects the image as the sample image and starts a round of active learning. Each service vehicle in the task group G will use their own local model to produce an annotation label based on their segmentation prediction of the sampled image. Then for each pixel n in the sampled image with total pixel number of N, the task vehicle conducts a label integration based weighted

[2]Cross entropy is defined as:

$$\mathcal{S}(\mathcal{I}) = \frac{1}{N}\sum_{n=1}^{N}\mathcal{H}(\phi_n) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C}\phi_{n,c}\log(\phi_{n,c}) \tag{6.1}$$

$\mathcal{H}(\phi_n)$ 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.

average as the following:

$$\mathcal{L}_n = \frac{1}{|G|} \sum_{v_k \in G}^{|G|} A_{v_k} \mathcal{L}_{k,n} \tag{6.2}$$

where $\mathcal{L}_n$ denotes the final annotation of pixel n decided by the task group G, $\mathcal{L}_{k,n}$ denotes the annotation of pixel n produced by vehicle k. If more than a specified threshold T_{th} of the pixels are labeled differently between the task vehicle's own prediction and the group annotation, the task vehicle will consider it as informative and add the sampled image to its training set. As discussed in [145], we update the task vehicle's local model once every 64 new sampled informative images to save resources and avoid the so-called "catastrophic forgetting" problem.

6.3.2 Task Group Formation

Let M denote the required number of service vehicles to form a task group G. A task group is formed when the task vehicle receives at least M feedback annotations from neighboring vehicles. If more than M feedback are received, the task vehicle sorts the feedback based on the rated accuracy of each service vehicle's local model and uses the M best feedback annotations for label integration. Note that every service vehicle that receives M feedback annotations from other group members could also add the sample to their training set if they find it informative. However, for evaluation purposes, we only study the behavior and results of the designated task vehicle. The detailed group formation procedure is provided in Algorithm 5.

6.4 Evaluation

To demonstrate how our proposed Sim2Scale co-simulation framework can benefit the evaluation and validation of CAV applications, the simulation results are evaluated by the following metrics:

- ***Model Prediction mIoU (mIoU):*** mean Intersection-Over-Union (IoU) of the per-

Algorithm 5: Task Group Formation

v_i denotes the task vehicle;
while v_i *has an informative image* $\mathcal{I}$ *as sample* **do**
    broadcast $\mathcal{I}$, local model's rated accuracy A_{v_i}, and v_i's annotation $\mathcal{L}_i$ of $\mathcal{I}$;
    initialize an empty group list $G = [\,]$;
    for *each vehicle* v_j *that received* v_i*'s data* **do**
        compute annotation labels based on v_j's local model;
        broadcast local model's rated accuracy A_{v_j}, and v_j's annotation $\mathcal{L}_j$ of $\mathcal{I}$;
    for *each* v_j *that* v_i *receives feedback from* **do**
        add v_j to G;
    if $|G| \geq M$ **then**
        select the $M-1$ received annotations with the highest A_{v_j} and compute the group annotation results based on Equation 6.2;
    if $|G| < M$ *and timeout* **then**
        send task drop notification to v_j in G;

class prediction performance, where the higher the mIoU achieves, the better the model performance is.

- ***Task Completion Rate (TCR):*** the percentage of completed tasks over the number of completed and failed tasks. A task is considered completed if the task vehicle successfully obtained integrated annotation from M service vehicles' feedback, while a task is considered failed if the task vehicle cannot aggregate enough feedback to annotate the sample image.

- ***Average Task Completion Time (aTCT):*** the average task completion time of all completed tasks of the task vehicle, where the task completion time for each task is calculated by the total simulation time duration between the task vehicle finding an informative sample and it deciding whether to add the sample to training set. Note that the model update (training) time is not counted in this metric.

Unless otherwise noted, following parameters were used as default setting in all experiments: number of vehicles $= 200$, message exchange frequency $= 50$HZ, the communication range between vehicles was set to 300m. Natural packet loss and delay were simulated such that messages were randomly dropped at receiving vehicles with a drop rate of 10%. The required number of service vehicles for a task group was set to $M = 3$. Data transmission rate in Veins follows the default 6 Mbps. A simulation run stops when the task vehicle successfully learned 512 informative image samples (512 rounds of successfully completed active learning).

6.4.1 Impact of the number of vehicles

We first evaluated how the number of vehicles affects the active learning results, as this parameter has a large impact on the traffic thus affecting whether the task vehicle can get enough required feedback labels to conduct label integration before it loses connection with the service vehicles. The number of vehicles was varied from 100 to 300 with an increment

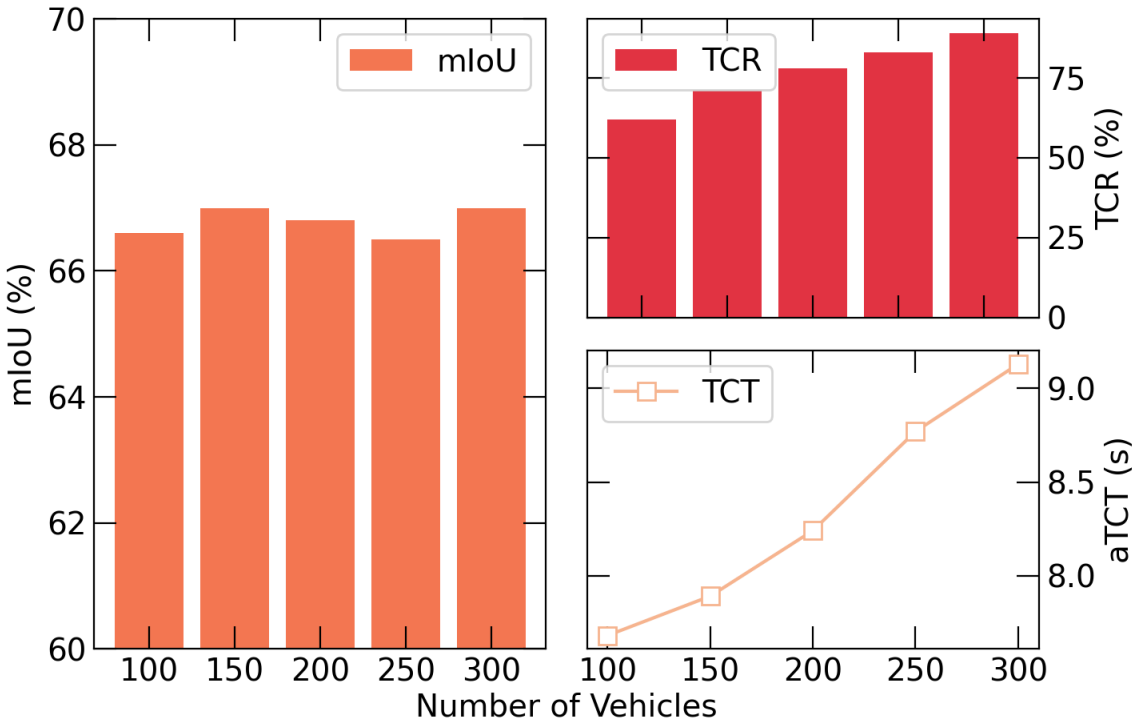


Figure 6.4: Performance vs. Number of Vehicles

of 50. As reported in Figure 6.4, as the number of vehicles increases, TCR increases from 62% - 89%, while the mIoU stays almost the same. As we stopped the simulation run after 512 informative samples were learned, the model performance, as indicated by mIoU, was not affected. However, the total number of attempted tasks was large when there were not enough vehicles, as indicated by the lower TCR. Moreover, aTCT increased slightly from 7.68s - 9.13s as the number of vehicles increased, which is likely due to a corresponding increase in the number of beacon messages sent. Without a co-simulation platform that includes simulation of a 3D scene, realistic traffic, and network communication together, it would not have been possible to carry out this level of detail in evaluation.

6.4.2 Impact of the task group size

Next, we varied the number of service vehicles M required to form a task group from 1 - 5 to study whether the size of the group affects the collaborative active learning performance. The results are reported in Figure 6.5. As the number of service vehicles increased, TCR

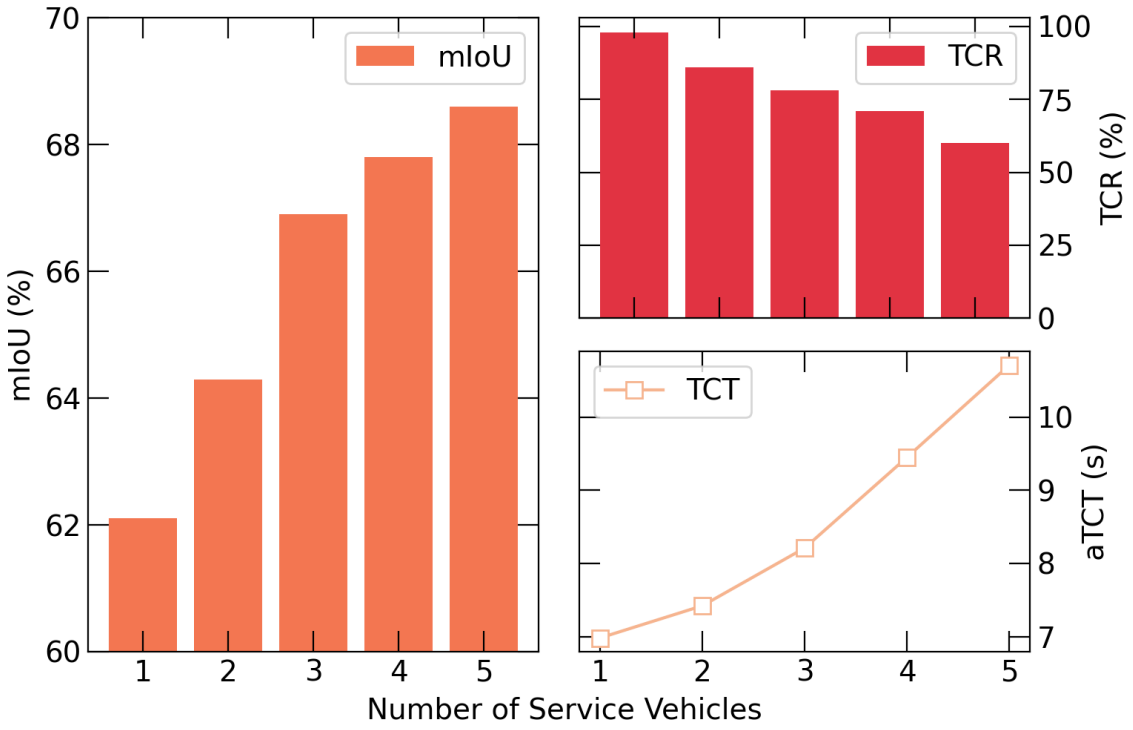


Figure 6.5: Performance vs. Number of Service vehicles

dropped from 98% to 61%, which indicates that the larger the task group required, the harder it becomes to complete the collaborative annotation task before the group breaks up. However, we also see that the mIoU increases from 62% - 68% as the number of service vehicles goes from 1 to 5. This aligns with results seen in related works [147] that the more participants, the better the training results become. Moreover, since more service vehicles in a group results in more data being exchanged, the aTCT increases as the number of service vehicles increases due to the increased communication time.

6.4.3 The impact of rounds of active learning

Typically, as the number of training images grows, better machine learning model is obtained. In addition, adding more rounds of active learning to obtain more informative training images should not largely affect the TCT and aTCT since workload for each round of active learning remains similar. To validate the assumptions, we varied the rounds of active learning from 128 to 896 with an increment of 128. As reported in Figure 6.6,

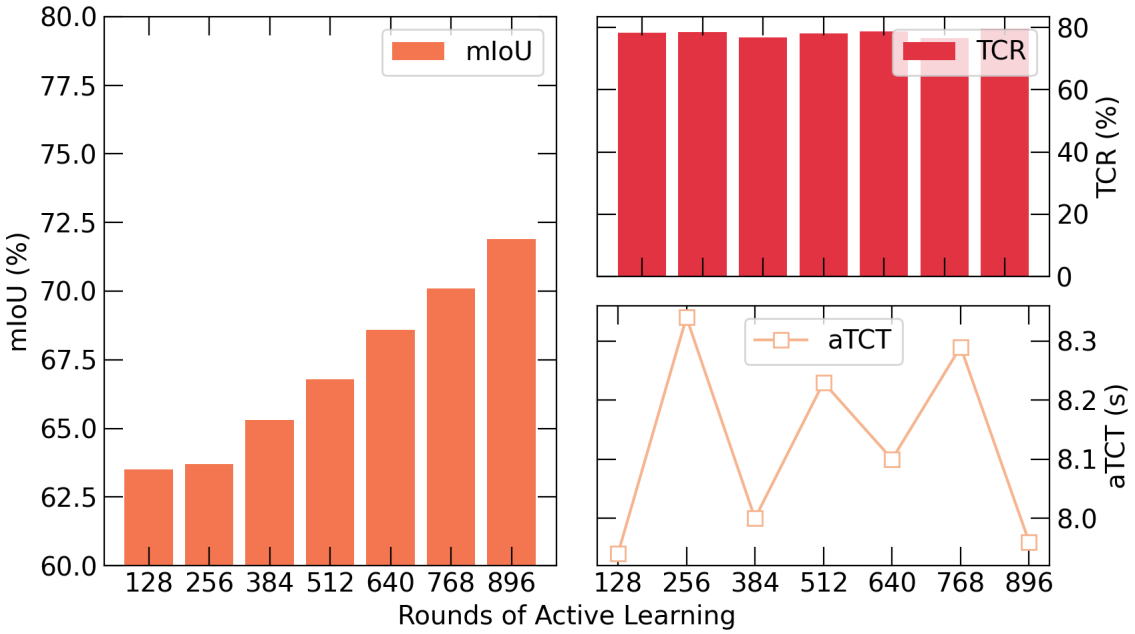


Figure 6.6: Performance vs. Rounds of Active Learning

when the rounds of active learning were below 384, there were too few data to improve the model performance. Once the training dataset grew large enough, the mIoU increased as the rounds of active learning increases. Moreover, TCR and aTCT varied in an ignorable range as the rounds of active learning changes, which verifies our assumption.

6.4.4 Discussion

The results presented in this section are not intended to provide a comprehensive evaluation of the performance of the example collaborative learning application. Rather, they are designed to provide a few examples of the types of evaluations that can be carried out with the Sim2Scale co-simulation framework. In particular, without integrating the realistic packet drop and delay models in Veins with CARLA and SUMO, it would not have been possible to accurately evaluate task completion rate and task completion time.

6.5 Chapter Summary

This chapter described Sim2Scale, a co-simulation framework built on top of open-sourced packages, which supports the evaluation and validation of CAV applications. It fills existing gaps between mobility/traffic, network communication, and 3D environment simulation. In addition, both GUI and Python script interfaces are provided for easy access. Evaluations of an example collaborative active learning algorithm demonstrated how Sim2Scale can be used to evaluate important performance metrics for CAV applications.

CHAPTER 7
CONCLUSIONS

7.1 Thesis Conclusions

As the irreversible trend of connected and autonomous vehicles, artificial intelligence and machine learning will continue to thrive and serve as the backbone of many CAVs applications, including perception, state estimation, probabilistic modeling, time series forecasting, gesture recognition, robustness guarantees, real-time constraints, user-machine communication, multi-agent planning, and intelligent infrastructure. However, the growth is hindered by the speed and cost of training data collection, as the better the coverage of high-quality data set of different scenarios, the better the model performance will be. Existing data collection process mainly relies on test vehicles driving on road and then offloading the recorded data to remote data centers for processing and labeling. Considering the variances on different road surfaces under different weather conditions, along with other vehicles' different reactions, the current method of data collection becomes the bottleneck for model accuracy. Besides, as the continual change of the external environment, relying on fixed pre-equipped machine learning models to make decisions will be far from sufficient. The model should be able to adapt to changes of external environment as the vehicle moves.

Therefore, in this thesis, we aimed to establish the framework foundation for the realization of a new learning framework – collaborative online active learning, where the vehicles can learn as they go, alleviating the stress of data collection and human annotation. The contribution of each chapter is summarized as follow,

- In Chapter 3, we investigate the problem of how to protect benign vehicles from making decisions based on received false information, as this is the ground of our

thesis objective – applying collaborative online active learning in vehicular network. Without information security, collaboration among vehicles will not be able to work. Thus, we introduce the concept of Proof-of-Eligibility Challenge, which limits the impact of compromised vehicles from outside of an event area by preventing them from participating in the consensus process. The Byzantine-fault-tolerant consensus algorithm for connected vehicles (BFCV) is presented to ensure information security among vehicles, without requiring privileged members, leader election, nor trusted shared key distribution. The algorithm also provides dynamic consensus group formation in an environment without a known pre-defined set of consensus participants. We report on the implementation of a BFCV prototype and simulation of it in a realistic environment built on top of Veins, SUMO and OMNet++. Evaluation results show that BFCV provides fast consensus satisfying both safety and liveness requirements.

- In Chapter 4, as completing computation-intensive tasks cooperatively among nearby vehicles in face of the highly mobile and transient vehicular network can hardly be achieved, a task-oriented group formation framework, based on the probability of successful task completion, is proposed. We show that task completion probability is primarily dependent on two random variables – the stay time of the vehicular task group and its task completion time. We use this framework and a notion of result quality to formulate the problem of selecting the best task group for a particular computation. We present a two-stage algorithm that performs task-oriented group formation for cooperative computations. The algorithm maximizes the size of groups in order to produce the best cooperative result while targeting a specified probability of task completion. We report on a prototype implementation of our group formation algorithm, which is based on distributed learning applications for autonomous vehicles. The prototype runs in a realistic environment built on top of Veins and SUMO. Evaluation results show that our algorithm achieves high task completion rates and

good group sizes across a wide range of traffic scenarios.

- In Chapter 5, to realize our thesis goal of learning without human annotator, a collaborative online annotation algorithm, which replaces human annotation by synthesizing the predictions generated by neighboring vehicles' local models and correlated multi-views is presented. As a result, data annotation and model updates can be completed locally without support of a centralized server. To facilitate the cooperative pseudo label generation process and improve pseudo label accuracy, a multi-view prediction transfer scheme is proposed to align different views from multiple vehicles via leveraging sensor data fusion. Besides, as no human labeler is needed, model updates can be completed locally without querying the centralized server, which can largely save communication resources. Additionally, 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.

- In Chapter 6, despite the appeal of the presented works in Chapter 3-5 and many other research works, testing and validations of these conceptual ideas and prototypes are of the same importance. Therefore, a co-simulation framework to develop, simulate, and evaluate algorithms and methodologies for collaborative computation tasks, which enables simulation of traffic, communication and 3D environment in connected and autonomous vehicles, is presented. Both Python script and QT GUI interfaces are provided for users with less familiarity with existing simulators to quickly and easily build and validate their ideas. An example application – a collaborative active learning is presented, which can be extended to other cooperative or

non-cooperative tasks with modest effort. The entire Sim2Scale framework and the collaborative active learning example application are open-source and freely available to the research community.[1]

7.2 Future Work

In this thesis, we focus on addressing the design and attest the effectiveness applying collaborative online active learning in the highly mobile vehicular network. Although best effort has been made to cover as many aspects of the topics as we can, there are still some remaining work for us to complete in the near future.

In Chapter 3, we have shown that the proposed "proof-of-eligibility" based byzantine-fault-tolerant consensus algorithm can achieve byzantine agreement with unknown group membership and unreliable communication channels and is targeted at vehicular network environments. However, two limitation shows in two aspects:

- *1)* the complete "proof-of-eligibility" puzzle design is very challenging. However to pre-invent enough puzzle problems which can provide sufficient proof of vicinity of in all situations remains a big challenge. However, enumerating all possible scenarios of the real world traffic can hardly happen, then, other advanced machine learning based methodologies may be studied to generate puzzles as the vehicle moves. Moreover, the effectiveness and efficiency of the "proof-of-eligibility" concept is heavily based on the accuracy of vehicles' equipped sensor sets. If the sensors fail, then a benign vehicle may not be able to successfully attest itself as being close-by, which limits its participation.

- *2)* to speed up the process of reaching consensus, we made a trade-off, where we parallel the process of reaching agreement of group membership and consensus decision – a decision is made if more than $2/3$ of the vehicles agree on the same decision and

[1] A demo is presented: *https://github.com/Jacquelinehy520/Sim2Scale* and code will be released once the paper submission notification come out.

on the group membership. This cannot guarantee that every member in the consensus group reach consensus on both group membership in the same time. Though this method does not let false information go through, it may also block certain correct information, which limits the efficiency.

In Chapter 4, we have demonstrated that our proposed task-oriented group formation algorithm has great potential in achieving high task completion while maximizing the group size. Nevertheless, it requires some prior knowledge of application-specific task completion time distributions. How to estimate and model the task completion time remains an interesting and important area to study. We will consider the following high-level ideas to approach this problem:

- *1)* identify the important variables of a task affecting the task completion time, such as rated computation speed, operations required for the task, communication cost, etc.;
- *2)* collect data through simulations under different traffic settings, and build a baseline statistical model for a specific algorithm (techniques such as survival analysis [148] may be applied);
- *3)* starting with the baseline model, apply online learning techniques [121] to tune the model during deployment.

In Chapter 5, as brought up in Section 5.6.3, we believe the group members' view diversity and the AL stage training set size are also interesting to study for future work. Due to time limitation, we cannot evaluate these parameters in detail. Future work will study the parameters discussed in 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.

7.3 Publications

As part of the research concluded in this dissertation, we have written several documents that are either published, submitted, or in progress as follow:

- H. Liu, C.-W. Lin, E. Kang, S. Shiraishi, and D. M.Blough, "A byzantine-tolerant distributed consensus algorithm for connected vehicles using proof-of-eligibility," in Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM), 2019.

- H. Liu and D. M. Blough, "Cooperative task-oriented group formation for vehicular networks," in 2022 IEEE 19th Annual Consumer Communications Networking Conference (CCNC), IEEE, 2022.

- H. Liu and D. M. Blough, "MultiVTrain: Collaborative multi-view active learning for segmentation in connected vehicles," in 2021 IEEE 18th International Conference on Mobile Ad Hoc and Smart Systems (MASS), 2021.

- H. Liu and D. M. Blough, "Sim2Scale: A Co-Simulation Framework for Developing and Evaluating Algorithms for Connected and Autonomous Vehicles," *to be submitted*, 2022.

REFERENCES

[1] H. Ye, L. Liang, *et al.*, "Machine learning for vehicular networks: Recent advances and application examples," *IEEE Vehicular Technology Magazine*, 2018.

[2] A. A. Abdellatif *et al.*, "Active learning-based classification in automated connected vehicles," *arXiv preprint arXiv:2002.07593*, 2020.

[3] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, 2020.

[4] *The KITTI vision benchmark suite*.

[5] K. Witrisal *et al.*, "Chapter 9 - localization and tracking," in *Inclusive Radio Communications for 5G and Beyond*, C. Oestges and F. Quitin, Eds., Academic Press, 2021, pp. 253–293, ISBN: 978-0-12-820581-5.

[6] NHTSA, *U.s. dot advances deployment of connected vehicle technology to prevent hundreds of thousands of crashes*, NHTSA Media, Ed.

[7] N. H. T. S. Administration, *Federal Motor Vehicle Safety Standards; V2V Communications, A Proposed Rule by the National Highway Traffic Safety Administration on 01/12/2017*. NHTSA, 2018.

[8] S. Abuelsamid, *Toyota has big plans to get cars talking to each other and infrastructure in the u.s.* Apr. 2018.

[9] *V2v safety technology now standard on cadillac cts sedans*, Mar. 2017.

[10] P. Apostolos and K. Alexey, *Cellular v2x as the essential enabler of superior global connected transportation services*, Jun. 2017.

[11] R. He *et al.*, "Propagation channels of 5g millimeter-wave vehicle-to-vehicle communications: Recent advances and future challenges," *IEEE vehicular technology magazine*, vol. 15, no. 1, pp. 16–26, 2019.

[12] C. R. Storck and F. Duarte-Figueiredo, "A survey of 5g technology evolution, standards, and infrastructure associated with vehicle-to-everything communications by internet of vehicles," *IEEE access*, vol. 8, pp. 117 593–117 614, 2020.

[13] Z. Zhou, H. Liao, *et al.*, "Reliable task offloading for vehicular fog computing under information asymmetry and information uncertainty," *IEEE Transactions on Vehicular Technology*, 2019.

[14] F. Sun *et al.*, "Cooperative task scheduling for computation offloading in vehicular cloud," *IEEE Transactions on Vehicular Technology*, 2018.

[15] S. Olariu, "A survey of vehicular cloud research: Trends, applications and challenges," *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[16] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, 2017.

[17] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, 2017.

[18] C. Yang, Y. Liu, *et al.*, "Efficient mobility-aware task offloading for vehicular edge computing networks," *IEEE Access*, 2019.

[19] G. Zhang, F. Shen, Y. Yang, H. Qian, and W. Yao, "Fair task offloading among fog nodes in fog computing networks," in *2018 IEEE International Conference on Communications (ICC)*, IEEE, 2018.

[20] C. Zhu, G. Pastor, Y. Xiao, Y. Li, and A. Ylae-Jaeaeski, "Fog following me: Latency and quality balanced task allocation in vehicular fog computing," in *2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, IEEE, 2018.

[21] W. Zhang, Z. Zhang, and H.-C. Chao, "Cooperative fog computing for dealing with big data in the internet of vehicles: Architecture and hierarchical resource management," *IEEE Communications Magazine*, 2017.

[22] M. LiWang, Z. Gao, *et al.*, "Multi-task offloading over vehicular clouds under graph-based representation," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, IEEE, 2020.

[23] J. Feng, Z. Liu, C. Wu, and Y. Ji, "AVE: Autonomous vehicular edge computing framework with aco-based scheduling," *IEEE Transactions on Vehicular Technology*, 2017.

[24] G. Qiao, S. Leng, and et al., "Collaborative task offloading in vehicular edge multi-access networks," *IEEE Communications*, 2018.

[25] Y. Sun *et al.*, "Adaptive learning-based task offloading for vehicular edge computing systems," *IEEE Transactions on Vehicular Technology*, 2019.

[26] J. Y. Yu and P. H. J. Chong, "A survey of clustering schemes for mobile ad hoc networks," *IEEE Communications Surveys & Tutorials*, 2005.

[27] R. Agarwal and D. Motwani, "Survey of clustering algorithms for MANET," *arXiv preprint arXiv:0912.2303*, 2009.

[28] M. Sood and S. Kanwar, "Clustering in MANET and VANET: A survey," in *2014 International Conference on Circuits, Systems, Communication and Information Technology Applications*, IEEE, 2014.

[29] C. Shea, B. Hassanabadi, and S. Valaee, "Mobility-based clustering in vanets using affinity propagation," in *GLOBECOM 2009-2009 IEEE Global Telecommunications Conference*, IEEE, 2009.

[30] C. Cooper, D. Franklin, M. Ros, F. Safaei, and M. Abolhasan, "A comparative survey of VANET clustering techniques," *IEEE Communications Surveys & Tutorials*, 2016.

[31] C. Allig and G. Wanielik, "Alignment of perception information for cooperative perception," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2019.

[32] Z. Li and C. Chigan, "On joint privacy and reputation assurance for vehicular ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 13, no. 10, 2014.

[33] F. Dotzer, L. Fischer, and P. Magiera, "Vars: A vehicle ad-hoc network reputation system," in *World of Wireless Mobile and Multimedia Networks, 2005. WoWMoM 2005. Sixth IEEE International Symposium on a*, IEEE, 2005.

[34] Q. Ding, X. Li, M. Jiang, and X. Zhou, "Reputation management in vehicular ad hoc networks," in *Int'l Conf. on Multimedia Technology*, IEEE, 2010.

[35] R. Heijden *et al.*, "Misbehavior detection in vehicular ad-hoc networks," *1st GI/ITG KuVS Fachgespräch Inter-Vehicle Communication*, 2013.

[36] M. Arshad, Z. Ullah, N. Ahmad, M. Khalid, H. Criuckshank, and Y. Cao, "A survey of local/cooperative-based malicious information detection techniques in vanets," *EURASIP Journal on Wireless Communications and Networking*, no. 1, 2018.

[37] T. Leinmüller, E. Schoch, F. Kargl, and C. Maihöfer, "Decentralized position verification in geographic ad hoc routing," *Security and communication networks*, vol. 3, no. 4, 2010.

[38] M. Ghosh, A. Varghese, A. Gupta, A. A. Kherani, and S. Muthaiah, "Detecting misbehaviors in vanet with integrated root-cause analysis," *Ad Hoc Networks*, vol. 8, no. 7, pp. 778–790, 2010.

[39] Z. Cao *et al.*, "Proof-of-relevance: Filtering false data via authentic consensus in vehicle ad-hoc networks," in *IEEE INFOCOM Workshops*, 2008.

[40] J. Petit and Z. Mammeri, "Dynamic consensus for secured vehicular ad hoc networks," in *Wireless and Mobile Computing, Networking and Communications*, 2011.

[41] L. Lamport *et al.*, "Paxos made simple," *ACM Sigact News*, vol. 32, no. 4, 2001.

[42] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm.," in *USENIX Annual Technical Conference*, 2014.

[43] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 4, no. 3, 1982.

[44] J. Augustine, G. Pandurangan, and P. Robinson, "Fast byzantine agreement in dynamic networks," in *Proceedings of the 2013 ACM symposium on Principles of distributed computing*, ACM, 2013.

[45] R. Guerraoui, F. Huc, and A. Kermarrec, "Highly dynamic distributed computing with byzantine failures," in *Proceedings of the 2013 ACM symposium on Principles of distributed computing*, ACM, 2013.

[46] M. Castro, B. Liskov, *et al.*, "Practical byzantine fault tolerance," in *OSDI*, vol. 99, 1999.

[47] D. Cavin, Y. Sasson, and A. Schiper, "Consensus with unknown participants or fundamental self-organization," in *International Conference on Ad-Hoc Networks and Wireless*, Springer, 2004.

[48] F. Greve and S. Tixeuil, "Knowledge connectivity vs. synchrony requirements for fault-tolerant agreement in unknown networks," in *Dependable Systems and Networks, 2007. DSN'07. 37th Annual IEEE/IFIP International Conference on*, IEEE, 2007.

[49] W. Wu, J. Cao, and M. Raynal, "Eventual clusterer: A modular approach to designing hierarchical consensus protocols in manets," *IEEE Transactions on Parallel & Distributed Systems*, no. 6, 2008.

[50] A. B., P. L., and F. G., "Solving consensus in opportunistic networks," in *ICDCN*, 2015.

[51] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[52] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," *self-published paper, August*, vol. 19, 2012.

[53] N. Kshetri, "Can blockchain strengthen the internet of things?" *IT Professional*, vol. 19, no. 4, 2017.

[54] A. Dorri, S. S. Kanhere, and R. Jurdak, "Towards an optimized blockchain for iot," in *Proceedings of the Second International Conference on Internet-of-Things Design and Implementation*, ACM, 2017.

[55] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "Blockchain for iot security and privacy: The case study of a smart home," in *Pervasive Computing and Communications Workshops (PerCom Workshops), 2017 IEEE International Conference on*, IEEE, 2017.

[56] S. Huh, S. Cho, and S. Kim, "Managing iot devices using blockchain platform," in *Advanced Communication Technology (ICACT), 2017 19th International Conference on*, IEEE, 2017.

[57] M. A. Walker, A. Dubey, A. Laszka, and D. C. Schmidt, "Platibart: A platform for transactive iot blockchain applications with repeatable testing," in *Proceedings of the 4th Workshop on Middleware and Applications for the Internet of Things*, ACM, 2017.

[58] *The 6 levels of vehicle autonomy explained*.

[59] *3 types of autonomous vehicle sensors*.

[60] Z. Li *et al.*, "User association for load balancing in vehicular networks: An online reinforcement learning approach," *IEEE T-ITS*, 2017.

[61] Z. Zhang, C. Sun, *et al.*, "Application of a machine learning method to evaluate road roughness from connected vehicles," *Journal of Transportation Engineering*, 2018.

[62] S. K. Tayyaba *et al.*, "5g vehicular network resource management for improving radio access through machine learning," 2020.

[63] W. Luo *et al.*, "Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net," in *IEEE CVPR*, 2018.

[64] Y. Wang *et al.*, "Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving," in *IEEE CVPR*, 2019.

[65] M. Ullah *et al.*, "Pednet: A spatio-temporal deep convolutional neural network for pedestrian segmentation," *Journal of Imaging*, 2018.

[66] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[67] M. Jaritz, R. De Charette, M. Toromanoff, E. Perot, and F. Nashashibi, "End-to-end race driving with deep reinforcement learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 2070–2075.

[68] H. Zhao *et al.*, "Icnet for real-time semantic segmentation on high-resolution images," in *Proceedings of the European Conference on Computer Vision*, 2018.

[69] J. Van Lint, "Online learning solutions for freeway travel time prediction," *IEEE Transactions on Intelligent Transportation Systems*, 2008.

[70] S. Shalev-Shwartz *et al.*, "Online learning and online convex optimization," *Foundations and trends in Machine Learning*, 2011.

[71] B. Settles, "From theories to queries: Active learning in practice," in *Active Learning and Experimental Design workshop In conjunction with AISTATS 2010*, 2011.

[72] E. Ferreira, A. R. Masson, *et al.*, "Adversarial bandit for online interactive active learning of zero-shot spoken language understanding," in *IEEE ICASSP*, 2016.

[73] D. Sculley, "Online active learning methods for fast label-efficient spam filtering.," in *CEAS*, 2007.

[74] A. Balakrishnan *et al.*, "Using contextual bandits with behavioral constraints for constrained online movie recommendation.," in *IJCAI*, 2018.

[75] S. C. Hoi, D. Sahoo, J. Lu, and P. Zhao, "Online learning: A comprehensive survey," *arXiv preprint arXiv:1802.02871*, 2018.

[76] Y. Yuan, S.-W. Chung, *et al.*, "Gradient-based active learning query strategy for end-to-end speech recognition," in *IEEE ICASSP*, 2019.

[77] D. L. Li *et al.*, "Multi-view vehicle detection based on fusion part model with active learning," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[78] A. Salem, "Stimulation and detection of android repackaged malware with active learning," *arXiv preprint arXiv:1808.01186*, 2018.

[79] X. Zhu, J. Lafferty, and R. Rosenfeld, "Semi-supervised learning with graphs," Ph.D. dissertation, Carnegie Mellon University, 2005.

[80] B. Settles, "Active learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2009.

[81] S. Kato *et al.*, "Autoware on board: Enabling autonomous vehicles with embedded systems," in *ACM/IEEE 9th ICCPS*, 2018.

[82] H. Veeraraghavan *et al.*, "Learning to recognize video-based spatiotemporal events," *IEEE Transactions on intelligent transportation systems*, 2009.

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

[84] D. Silver *et al.*, "Active learning from demonstration for robust autonomous navigation," in *2012 IEEE International Conference on Robotics and Automation*, 2012.

[85] K. Satzoda *et al.*, "Multipart vehicle detection using symmetry-derived analysis and active learning," *IEEE Transactions on Intelligent Transportation Systems*, 2016.

[86] S. Das *et al.*, "Multi-criteria online frame-subset selection for autonomous vehicle videos," *Pattern Recognition Letters*,

[87] M. K. Abdel-Aziz *et al.*, "Ultra-reliable and low-latency vehicular communication: An active learning approach," *IEEE Communications Letters*, 2019.

[88] H. Alghodhaifi and S. Lakshmanan, "Autonomous vehicle evaluation: A comprehensive survey on modeling and simulation approaches," *IEEE Access*, 2021.

[89] *California connected vehicle testbed*, caconnectedvehicletestbed.org.

[90] *Intelligent transportation systems - Ann Arbor connected vehicle test environment (AACVTE)*.

[91] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO-simulation of urban mobility," *International Journal on Advances in Systems and Measurements*, 2012.

[92] S. Aoki and R. Rajkumar, "A merging protocol for self-driving vehicles," in *2017 ACM/IEEE 8th International Conference on Cyber-Physical Systems (ICCPS)*, 2017.

[93] G. Cecchini, A. Bazzi, *et al.*, "LTEV2Vsim: An LTE-V2V simulator for the investigation of resource allocation for cooperative awareness," in *2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, 2017.

[94] R. Mangharam, D. Weller, R. Rajkumar, P. Mudalige, and F. Bai, "Groovenet: A hybrid simulator for vehicle-to-vehicle networks," in *2006 Third Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services*, 2006.

[95] C. Sommer, R. German, and F. Dressler, "Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis," *IEEE Transactions on Mobile Computing*,

[96] M. Piorkowski, M. Raya, A. L. Lugo, P. Papadimitratos, M. Grossglauser, and J.-P. Hubaux, "TraNS: Realistic joint traffic and network simulator for VANETs," *ACM SIGMOBILE mobile computing and communications review*, 2008.

[97] T.-K. Lee, T.-W. Wang, W.-X. Wu, Y.-C. Kuo, *et al.*, "Building a v2x simulation framework for future autonomous driving," in *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 2019.

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

[99] S. Aoki, L. E. Jan, J. Zhao, A. Bhat, R. R. Rajkumar, and C.-F. Chang, "Co-simulation platform for developing inforich energy-efficient connected and automated vehicles," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, 2020.

[100] C. Qi and M. Z. Morley, *Connected cars can lie, posing a new threat to smart cities*.

[101] Q. Chen, Y. Yin, Y. Feng, Z. M. Mao, and H. Liu, "Exposing congestion attack on emerging connected vehicle based traffic signal control,"

[102] M. Al-Kahtani, "Survey on security attacks in vehicular ad hoc networks (vanets)," in *Signal Processing and Communication Systems (ICSPCS), 2012 6th International Conference on*, IEEE, 2012.

[103] R. Heijden *et al.*, "Survey on misbehavior detection in cooperative intelligent transportation systems," *arXiv preprint*, 2016.

[104] N. Lo and H. Tsai, "Illusion attack on vanet applications-a message plausibility problem," in *Globecom Workshops, 2007 IEEE*, IEEE, 2007.

[105] J. Grover *et al.*, "Machine learning approach for multiple misbehavior detection in vanet," in *International Conference on Advances in Computing and Communications*, 2011.

[106] NHTSA, *Federal motor vehicle safety standards; v2v communications*, NHTSA Media, Ed.

[107] J. Douceur, "The sybil attack," in *International workshop on peer-to-peer systems*, Springer, 2002.

[108] K. Rabieh, M. Mahmoud, T. N. Guo, and M. Younis, "Cross-layer scheme for detecting large-scale colluding sybil attack in vanets," in *2015 IEEE International Conference on Communications (ICC)*, IEEE, 2015.

[109] Y. Yao *et al.*, "Multi-channel based sybil attack detection in vehicular ad hoc networks using rssi," *IEEE Transactions on Mobile Computing*, vol. 18, no. 2, 2019.

[110] T. Garip, H. Kim, P. Reiher, and M. Gerla, "Interloc: An interference-aware rssi-based localization and sybil attack detection mechanism for vehicular ad hoc networks," in *14th Annual Consumer Comm. & Networking Conf.*, IEEE, 2017.

[111] L. Wu and H. Tsai, "Modeling vehicle-to-vehicle visible light communication link duration with empirical data," in *Globecom Workshops, 2013*, IEEE, 2013.

[112] OpenStreetMap contributors, *Planet dump retrieved from https://planet.osm.org*, https://www.openstreetmap.org, 2017.

[113] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO - Simulation of Urban Mobility," *International Journal On Advances in Systems and Measurements*, Int'l Journal On Advances in Systems and Measurements, vol. 5, no. 3&4, Dec. 2012.

[114] NHTSA. "Federal motor vehicle safety standards; v2v communications." (2017).

[115] S. Dmitriev, *Autonomous cars will generate more than 300 tb of data per year*, Jul. 2020.

[116] A. Alhilal, T. Braud, and P. Hui, "Distributed vehicular computing at the dawn of 5g: A survey," *arXiv preprint arXiv:2001.07077*, 2020.

[117] G. Wang, J. Hu, Z. Li, and L. Li, "Cooperative lane changing via deep reinforcement learning," *arXiv preprint arXiv:1906.08662*, 2019.

[118] H. Liu, C.-W. Lin, E. Kang, S. Shiraishi, and D. M. Blough, "A byzantine-tolerant distributed consensus algorithm for connected vehicles using proof-of-eligibility," in *Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2019, pp. 225–234.

[119] H. Liu and D. Blough, "MultiVTrain: collaborative Multi-View active learning for segmentation in connected vehicles," in *2021 IEEE 18th International Conference on Mobile Ad Hoc and Smart Systems (MASS) (IEEE MASS 2021)*, 2021.

[120] NVIDIA Corporation. "NVIDIA drive AGX developer kit." (2020), (visited on 09/01/2020).

[121] 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.

[122] M. Ren, L. Khoukhi, H. Labiod, J. Zhang, and V. Veque, "A new mobility-based clustering algorithm for vehicular ad hoc networks (VANETs)," in *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*, IEEE, 2016.

[123] A. Daeinabi *et al.*, "VWCA: An efficient clustering algorithm in vehicular ad hoc networks," *Journal of Network and Computer Applications*, 2011.

[124] Y. Huo *et al.*, "An enhanced low overhead and stable clustering scheme for crossroads in VANETs," *EURASIP JWCN*, 2016.

[125] M. Ren, J. Zhang, and etl al., "A unified framework of clustering approach in vehicular ad hoc networks," *IEEE Transactions on intelligent transportation systems*, 2017.

[126] K. Bonawitz, H. Eichner, and et al., "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

[127] Z.-H. Zhou and M. Li, "Tri-training: Exploiting unlabeled data using three classifiers," *IEEE Transactions on Knowledge and Data Engineering*, 2005.

[128] M. Ren, L. Khoukhi, and et al., "A mobility-based scheme for dynamic clustering in vehicular ad-hoc networks (VANETs)," *Vehicular Communications*, 2017.

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

[130] G. L. Oliveira *et al.*, "Efficient deep models for monocular road segmentation," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

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

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

[133] 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.

[134] 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.

[135] 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.

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

[137] 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.

[138] 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.

[139] 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.

[140] 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.

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

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

[143] *Waymo self-driving test vehicle hits pedestrian but Waymo says a human was driving*.

[144] H. Liu and D. Blough, "Cooperative task-oriented group formation for vehicular networks," in *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*, 2022.

[145] H. Liu and D. M. Blough, "MultiVTrain: Collaborative multi-view active learning for segmentation in connected vehicles," in *2021 IEEE 18th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*, 2021.

[146] OMNeT++.

[147] H. Liu and D. M. Blough, "Cooperative task-oriented group formation for vehicular networks," in *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*, IEEE, 2022, pp. 584–592.

[148] J. Wang *et al.*, "Estimating the completion time of crowdsourced tasks using survival analysis models," *Crowdsourcing for Search and Data Mining*, 2011.